百问 FreeSwitch
（第二版）

余洪涌 编著

2014 年 9 月 中国厦门

文档历史:

版本号	日期	描述
1.0	2012-11-10	yhy 建立文档
1.1	2013-06-11	yhy 补充 windows 下的 PJSIP 软电话和安卓下软电话 ImsDroid 的编译和单机最大支持多少线并发通话
1.2	2013-06-15	yhy 补充使用 sipp 进行对 FreeSwitch 进行压力测试
1.3	2013-06-23	yhy 整理 NAT 穿透部分，独立出来
1.4	2013-07-13	yhy 整理 FreeSwitch 媒体部分，增加使用 mysql 作为 FreeSwitch 工作数据库
1.5	2013-07-19	yhy 增加 FAX 和 SRTP
1.6	2013-07-21	yhy 增加异常测试和 CTI 平台开发
1.6	2013-07-23	yhy 增加转码测试
1.7	2013-08-06	yhy 完成一轮校对工作
1.8	2013-08-28	yhy 增加 Sangoma 接口卡章节
1.9	2013-11-04	yhy 增加 IMS 接入支持章节,增加 ESL 的 ASR/SVR 开发,32 位 OS 支持超过 2G 内存，通话超时挂机，如何真正启用 VAD,uuid_bridge 说明等内容
2.0	2014-08-18 第二版	yhy 增加 FS1.4.7 版本的 WebRTC 接入，支持章节。 更新 FS 版本到 1.2.23，修正许多错误的问题回答。 修正许多笔误，去掉一些在新版本下已经不需要的问题部分。 增加 FS 高级设置一些新问题部分，比如落地的讨论。 补充完善了 CTIAPI 平台部分,增加空号检测模块。

版权声明：

之前第一版已经声明：本书没有版权，即使有也是属于互联网屌丝们的。
第二版依旧延续第一版的版权模式：
电子版大家随便看，随便传阅，只要不瞎改，不卖钱牟利就行，
特别是不要当作盈利目的商业用途：比如打印出来，印刷出来出售。
这个要求应该不算高。
书的印刷版可以通过下面两种方式购买：
1.通过万能的淘宝购买，淘宝地址：http://item.taobao.com/item.htm?id=37104577378
2.直接转账到本人的支付宝帐号: 13606060253，然后 qq 或短信通知本人收货人的地址姓名。

序：

这个通常需要一个牛 X 人物来写，还没有找到合适的。因为偶认识的牛人都不是做这个的，做这个的又都不是牛人。

第二版依然没有序，估计将来也不会有序。

联系作者：

name:余洪涌

tel：13606060253（早 9 点-晚 9 点，其他时间不建议联系）

qq：109559405 （随时可以联系，不在线就留言）

mail：109559405@qq.com

微博：暂无

微信：暂无

作者简介：

余洪涌：1974--?,国家系统分析员，从 1997 年起一直从事语音在电信行业的应用方面计算机系统研发工作。曾经开发过多个大型电信平台，实网使用用户达上百万,单系统并发呼叫用户达数千线。

从事过 D 卡，N 卡，东进等语音板卡开发，也从事过 dialogic HMP，毅航 ISX，东进 Keygoe，FreeSwitch 等多媒体交换机设备的开发工作。曾经在中科院声学所,中科信利国家重点语音实验室学习和工作多年，对计算机语音端点检测，语音编解码，语音识别，声纹识别,VOIP 及其在实际项目中的应用等有深入的了解。曾在国家核心期刊上发表一篇论文，拥有两项发明专利。主要使用的开发工具：VC6，VS2010，Eclipse，Flash Builder。

于 2010 年年底某天顿悟:在中国当前的房子价格面前，一切都是浮云。

在 2011 年与恒信移动合同到期之后，自觉退休在家带孩子。

目录

前言：

照惯例，要写一下为啥写本书。说白了纯粹是因为爱好，或者是吃饱了撑着写的。

公元 2012，玛雅的世界末日前的几个月，大约 2012 六月是偶开始接触 FreeSwitch。加入国内的 FreeSwitch 的群：190435825。

由于做过几年的其它 VOIP 产品的开发，比如：dialogic HMP，东进 Keygoe，毅航 ISX 系列的 VOIP 产品的开发。对于 SIP 也算是稍微有些了解，但是折腾 FreeSwitch 的过程对我来说也是一条崎岖的山路，一路折腾着过来。

在 FreeSwitch 群里面经常回答一些同学碰到一些怎么样才能达到啥啥功能的问题。估计也是一样在痛苦着。假如有人告诉它们要解决怎么样问题需要怎么办，同学们就不用那么折腾。

实际上大多的问题一来是因为不了解 SIP 的一下应用的基础知识，比如被 SDP 搞晕，不知道如何抓包，二来是因为 FreeSwitch 配置文件巨多。

碰到实际的问题不知道需要如何配置 FreeSwitch 的问题，如何分析 SIP 信令的问题。有些问题比较低级，有些问题比较高级。比如最经典的问题是回答“怎么知道谁有没有注册在线？”

每次回答的结果是一样的：sofia status profile internal reg XXXX

这个命令又臭又长的命令，我每次都记不住。每次都是从一个记事本 readme.txt 里面拷贝。林林总总回答的不厌其烦。归根到底是因为 FreeSwitch 的功能太牛 X，导致配置文件太多。直接统计 Conf 目录下的 XML 文件就达到接近 200 个配置文件。

初学者碰到问题要找到哪个配置文件，修改哪个配置参数，实在是一个让人抓狂的事情。更加不用说有些功能需要修改几个配置参数的组合情况。平时自己做研发事情的习惯是带着问题去研究，有问题有笔记，于是后面就把这些问题归类整理，起名《百问 FreeSwitch》。

第一个版本大约在世界末日前一个月提交到国内的 FreeSwitch 群的群共享里面，随后又不断增加问题，不断增加内容，陆续提交了几个版本。

文档虽然很早就放到 FreeSwitch 群共享里面供同学们交流参考。不过貌似初学者同学们大多还是不知道群共享文档，于是许多问题的回答答案变成了：“群共享里面有百问文档，可以参考。”

这样的好处是大多人只要回答一次，以后就自己看文档了，问题相对就少了，也算没有白写。

实际上许多问题的答案在 http://wiki.freeswitch.org 上都能找到，但是由于 WIKI 的设计不是针对问题的回答。你要是带着问题去寻找答案，那过程将是非常漫长和痛苦的，加上因为 WIKI 是英文的，更加加大了找到问题的过程，有些同学们英文不太过关，看起来就更加累。这里面写的很多问题都是同学们根据时间需要自己提出的，也有一些是我根据之前的一些经验列出的。

本书还有一大部分根据现实项目的需要改动或者说改造了 FreeSwitch 的一些代码。毕竟 FreeSwitch 是开源的东东，能够根据实际项目的需要进行改造，才能充分体现出开源的优势，但是反过来改动是一把双刃剑，不到万不得以不建议改动 FreeSwitch 的代码，因为改动后意味着将来的版本升级会非常麻烦。

不知道在哪里看见一句话：“程序员是运动员，不是理论家。”党也告诉我们实践是检验真理的唯一标准。碰到问题只有自己创造测试环境，测试条件，动手实践，测试，问题才能算真正解决。

话说回来，自己搞 FreeSwitch 也没有多久，勉强也只是入门而已。许多模块从未涉猎，许多问题回答不了。另外由于水平和条件限制写这个文档疏漏错误也是难免，假如各位同学有发现错误，请 mail：109559405@qq.com 偶会马上对错误进行确定以及修正。

某天看到林语堂先生写的一些关于写作描述，很有感触，补充记录如下：

“用随意文体的作家是以真诚的态度说话。他把他的弱点完全显露出来，所以他是从无防人之心的。作家和读者之间的关系，不应像师生的关系，而应像厮熟朋友的关系。

只有如此，方能渐渐生出热情。

凡在写作中不敢用“我”字的人绝不能成为一个好作家。

。。。。。。。。

各地方的菜馆大小不一，有些是高厅大厦，金碧辉煌，可设盛宴；有些是专供小饮。我所最喜欢的是同着两三个知己朋友到这种小馆子里去小饮，而极不愿意赴要人或富翁的盛宴。我们在这小馆子里边又吃又喝，随便谈天，互相嘲谑，甚至杯翻酒泼，这种快乐是盛席上的座客所享不到的，也是梦想不到的。

。。。。。。。。。”

作为前言。

第二版前言：

第一版完成之后放到群共享里面，许多同学下载之后发现许多笔误和错误，在此再次表示感谢。

还有许多同学购买了印刷版的，在此也再次表示感谢。在第一版初版之后，陆陆续续更新补充了许多问题。每次有补充，我都会发到群共享里面，遗憾的是很多新进群的伙计总是不知道到共享里面下载。

总体来说本书推出第二版的缘由大约有以下这些：

本书第一版出版到现在已经 1 年多了。

本书的版本更新比较快，不过 FreeSwitch 版本更新更快。

1.2.X 到现在已经到 1.2.23 的版本。

1.4.X 的也发布正式版本到 1.4.7 版本。

1.4.7 的版本开始支持 WebRTC。官方现在开始建议投产系统使用 1.4.7 的版本。

一些问题和回答随着 FreeSwitch 版本的更新现在已经过时需要全面整理。

在实际使用中，新碰到的一些具体问题以及群里面的同学们新提出的实际问题补充到 FreeSwitch 高级设置功能。

以上这些林林总总，便是本书推出第二版的缘由 **J**。

致谢：

感谢 CCAV，感谢 GOOGLE，感谢百度，感谢 FreeSwitch 群和群里面的同学，感谢中科信利，

感谢 FreeSwitch.org，感谢 wiki.FreeSwitch.org，感谢 FreeSwitch.org.cn

感谢 ~！@#￥%……&*（）—+

声明：

首先,需要声明，本文档不是严谨的学术书籍，是针对实际应用研发和部署使用过程中的一些问题的总结，很多的问题答案只是一种解决办法。

我相信许多答案一定不是最佳答案，很多答案只是算凑合能解决的办法之一。

其次,特别要指出的是本文档不是回答 WHY 的，大多是回答 HOWTO 的。

因此假如你想要知道 WHY，我认为作为程序员，没有其它途径，直接看 code 是最快的办法。因为本文档不是 FreeSwitch 的源码分析文档，因此回答不了 WHY。

最后,由于环境的不同，问题的答案甚至在各位读者的测试环境中都无法重现。甚至是错误的答案。:-）

因此，本书尽量描述清楚问题的出现环境。
本书的测试环境是大多是基于 FreeSwitch 的 windows 版的 1.2.1，1.2.5.3，1.2.7，1.2.10...1.2.23,以及 1.4.7 的源码上编译的。很少一部分会提到 linux 下的版本，因此假如你是 linux 下的开发，请注意 windows os 和 linux os 的区别，比如盘符， / 与 \的区别等等。
假如你碰到一样的问题，但是出现的情况跟文档列出的不一致，欢迎通过上面的联系方式进行沟通。或者加入到 百问 FreeSwitch 群里面进行交流，百问 FreeSwitch 群号是：328024052。

适合的本文档读者对象：

毫无疑问 FreeSwitch 百问的读者肯定是技术人员，他们可能是：

A. 对 VOIP 有兴趣没有基础的伙计
B. 对 FreeSwitch 有兴趣站在门口观望的伙计，这个文档可以解决你的许多困惑
C. 计划把 FreeSwitch 从实验所或者研发中心部署应用到实网系统中的伙计
D. 准备使用 FreeSwitch 做 IPPBX 的伙计
E. 准备开发软交换或者 VOIP 呼叫中心的伙计
F. 对 FreeSwitch 进行运营维护的伙计
G. 之前搞 Asterisk，由于受到各种困扰，现在准备跳槽搞 FreeSwitch 的同学

大体说来，本文档比较适合与搞软交换或者 FreeSwitch 初级和中级研发运营工程人员。

不适合的本文档读者对象：

本人上面说过百问 FreeSwitch 不是解决 WHY 的学术著作，因此以下这些人我认为不合适看，
他们可能是：

A. 搞 VOIP 研究的学校里面的老师和同学
B. FreeSwitch 高手，这些人通常都是直接改源码
C. 其他的牛人

假如要看在看完岛国 A 片之后没事做随便浏览看看免费的电子文档还行，浪费钱买印刷版的书就不必了。

本文档涉及的环境和工具：

首先当然各种硬件：

一个 N 年前的 E10 的笔记本，安装各种软电话软件，为了编译 PJSIP 安装了 VS2008。

另外一个是 K29 乞丐版（B950CPU，2G 内存，320G 硬盘），万能的某宝上花了 RMB2650 买的，然后就是一段折腾的道路：加了 4G 内存，换了从 TW 带回来了一个 M5P 128G 的 SSD 固态硬盘，最后再换了一个 3612QM 的 CPU，最后到今年硬盘不够，又增加了一个从某东上买的 M6M 的 mSATA 128G 固态硬盘。（跟大多数同学不一样，我的笔记本硬盘容量不够绝对不是因为 A 片放多了导致硬盘不够用，而是因为 FS 的一个版本编译之后差不多去掉 1-2G 硬盘空间，机器上保存非常多 FS 的版本结果就是硬盘不够。）

搞这么多纯粹是为了不想在编译 FreeSwitch 的时候等到花儿也谢了,现在重新编译一遍也只是分分钟的事情。这个机器安装的是 WIN7 32 位版本+VS2010 编译器，为了编译 FreeSwitch。

一个是 Dell R720 服务器 E5-2640*2 CPU 16G 内存 500G 硬盘*2 做 raid1 硬盘，用来做单机千线的压力和性能测试。

亿联的 T20/T20P IP 话机，

迅时的 MX8 VOIP 网关，

三星的安卓手机 NOTE 和安卓 PAD，主要是安装测试安卓版本的软电话。

Dell 11 PRO win8 的 pad，作为测试客户视频使用。不过尽然也可以作为服务器跑，跑 fs 也能支持很多线并发。

中兴红牛 V5 手机，测试 TD-LTE 用的。由于没有开通套餐，测试网速用掉了移动送的 150MB 不算还超出了几百 MB，结果移动算 1MB = RMB 1 元简直就是杀猪价。测试几把网速，等收到流量告警短信通知已经欠费几百了。

然后当然是 FreeSwitch 的源码，各种版本 1.2.1 1.2.5.3 1.2.7，一直到最新的稳定版本 1.2.23 和 1.4.7，由于吃了一次 git 最新版本的教训，偶决定都是下载 tar 好的稳定版本进行编译。

然后就是各种各样的五花八门的软电话，eyebeam(商业版的 Xlite)，kapanga，linphone，ImsDroid，MicroSip 还有开源的 pjsip.包括 windows 版本和安卓版本。

最后就是各种抓包工具。

以上的这些软电话和 7788 的工具可以 ftp 匿名到 42.120.20.89 上下载。

本文档约定：

五号下划线：URL，sip 地址，mail 地址等。比如 http://www.zeroc.com/ice.html

五号斜体：文档强调的部分，比如配置，日志的输出需要关心的部分等。

五号字体：正常文档。

小五号字体：代码和日志，配置文件参数，接口文档啥的。

本文档涉及内容：

本文档涉及的内容有 voip 基础，FreeSwitch 基础。

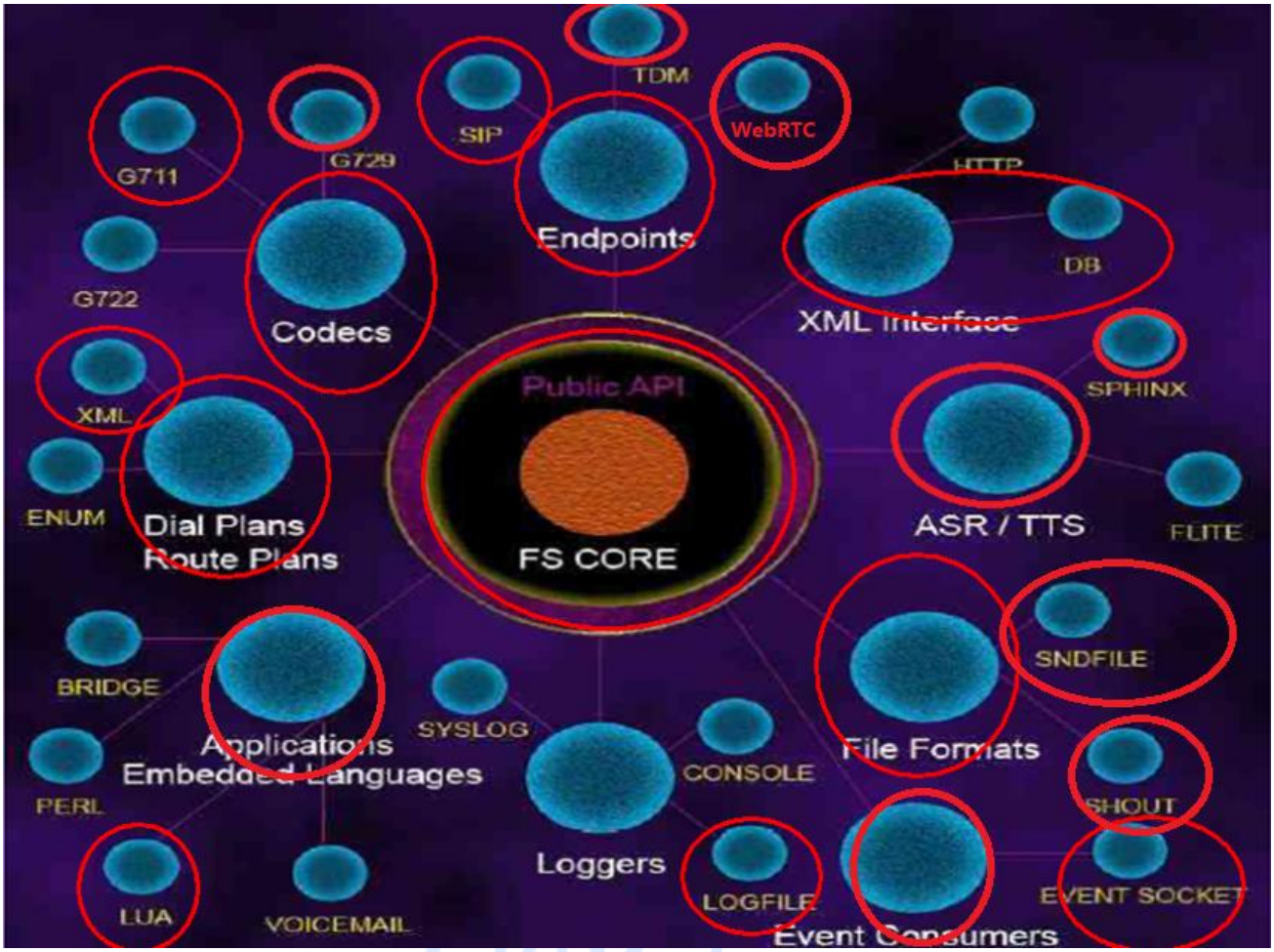
只有很少一部分 FreeSwitch 的源码以及修改（不到万不得已，我不想去修改源码，否则将来升级每次都得再修改一遍，这个感觉就像觉得自己之前白活了一段时间）。

FreeSwitch 的整个软件构建比较复杂，如下图：

本文档没有涉及全部模块。

涉及 FreeSwitch 的部分如下图 FreeSwitch 的整个软件构建的里面红色圈圈内的部分:

（这样的好处是，每当偶有增加内容就再圈上一块，呵呵）



当然还有许多是跟 FreeSwitch 无关的部分，比如 VOIP 基础，比如 CTIAPI 平台开发等等。
一个是搞 FreeSwitch 的基础，一个是搞 FreeSwitch 平台扩展，方便将来的业务开发。
下面先大概介绍文档的章节：

第一章 VOIP 基础部分
主要介绍一些 VOIP 必备的基础知识
以及一些常用的软电话和抓包之类的工具软件，
假如你之前做过 VOIP，可以直接跳过。
第二章 VOIP NAT 穿透部分
主要介绍 NAT 的一些知识，很多问题都是因为 NAT 导致没有声音或者
单通引发的。假如你对 NAT，很溜，可以直接跳过。
第三章 FreeSwitch 基础和配置部分
主要介绍 FreeSwitch 一些基本的配置，sip 和拨号计划。适合菜鸟入门。
第四章 FreeSwitch 高级配置部分
主要介绍 FreeSwitch 一些应用级别的配置，比如如何实网部署。
第五章 FreeSwitch 性能测试部分
主要介绍如何对 FreeSwitch 进行性能和压力测试，以及如何优化。
第六章 FreeSwitch 异常测试部分
主要介绍如何跟对 FreeSwitch 过不去，如何玩死它。
第七章 FreeSwitch FlashPhone 部分

主要介绍如何使用 FreeSwitch 跟 FlashPlayer 结合作为浏览器里面的软电话，
在 webrtc 没有完善发布之前，使用 flash 这个是一个解决办法。
第八章 FreeSwitch IM 消息聊天部分
主要介绍如何使用 FreeSwitch 作为聊天服务器，以及如何实现漏话消息通知。
第九章 FreeSwitch 媒体部分
主要介绍如何使用 FreeSwitch 各种媒体播音录音，会议，传真和加密语音
第十章 FreeSwitch ESL 编程部分
主要介绍如何通过 ESL 控制使用 FreeSwitch 里面的行为，实现自己的 IVR 流程
第十一章 CTI 平台开发部分
主要介绍如何设计开发一个功能完备 CTI 的平台，
对那些在熟悉 ESL 之后，要进一步提升自己的同学，可以认真看，特别是里面的
系统软件架构设计跟 CTI API DLL 设计部分。这个平台架构是本人多年积累拼凑出来的。
第十二章是 FreeSwitch 接口卡 Sangoma 部分
FreeSwitch 接口卡 Sangoma 部分主要介绍 Sangoma 的安装配置和使用，对于那些需要落
地的系统是需要知道的。
第十三章是 IMS 部分
IMS 章节部分主要是介绍如何使用 IMS 进行落地。包括使用 IMS 接入以实现用户拨入，和使用 IMS 进行系统主动拨出到用户。另外还介绍了 IMS 的一些特殊的要求，比如号码是 86 开头的等等。
第十四章是 WebRTC 部分
第十五章是空号检测模块部分，适合于电话营销系统的群呼的空号检测。

最后一部分是附录
列出一些参考资料，以及一些本人碰到的 bug 和本文档相关代码的下载地址。
废话少说，下面开始正文。

第一章 VOIP 基础部分

0. 研究 VOIP/FreeSwitch 之前需要哪些基础知识？

假如你不开发应用，只是使用 FreeSwitch，通常熟悉 SIP 就够。

假如要你开发应用，熟悉 C/C++是必须的。虽然 FreeSwitch 支持很多语言的开发接口，但是实际上基本都是在 C/C++的接口上通过 SWIG 进行扩展的。通过 swig 包装支持多种脚本语本语言控制呼叫流程,如 perl,php,lua,python,ruby,java,tcl 等。

事件套接字使用 Event Socket 可以使用任何其它语言通过 Socket 方式控制呼叫流程、扩 展 FreeSwitch 功能。

通俗讲 FreeSwitch 是也是一个 VOIP，目前主流的 VOIP，甚至包括运营商的 IMS 毫无例外都是基于 SIP 协议的。因此要搞定 FreeSwitch，SIP 协议是必须的。

1. VOIP 基础设施有哪些？

以太网交换机或者 HUB，这个是废话，没有这咋实现 IP 通信呢。

IP 话机,相当于普通电话机，不过不是普通电话机的电话线 RJ11 的接口 而是网线 RJ45 接口。

软电话，安装在电脑上的（通常是 windows 下）模拟 ip 话机的软件。当然安卓和 linux 下也有软电话。

SIP Server/IPPBX，完成 VOIP 交换的设备或者安装在 pc 上的软件。

IP 网关，通常是完成 VOIP 网络 到 PSTN E1 数字接口或者模拟电话线接口转换的设备。

2. SIP 基本问题有哪些？

SIP 协议广义上是包括 SIP 协议 ，RTP 协议和 SDP 协议。

它们的关系可以这样描述：

SIP 是信令控制，相当于电信里面的 7 号信令，或者相当于发号施令的工头领导角色，SIP 通常只有一个端口 5060 就可以工作。

RTP 是媒体传输，相当于干体力活的民工或者 IT 程序员角色，工头或者项目经理一句话，民工或者 IT 程序员就要忙几天几夜。RTP 通常需要许多端口来支持不同的线路。

通常是一个端口的区间，比如 10000-20000 端口区间。

SDP 是媒体描述，相当于描述使用哪些或者哪种语音和视频编码进行协商,然后使用协商的结果进行沟通。打个不恰当的比方：比如项目经理让某个 IT 程序员使用 C 语言开发，让另外一个 IT 程序员使用 java 语音开发，它们之间如何沟通呢？讨论的结果可以使用中间件比如 ICE，也可以使用 JNI.这个讨论的过程就是 SDP。

SDP 最扯淡的核心问题就在于是讨论的。就像你去买鞋一样，通常要试穿几次，合不合脚。假如不试穿就买了，结果又不合脚，很难谈清楚是你的脚有问题还是鞋有问题。

举例说明：a 用户使用电话打给 b 手机

SIP 是信令控制 相当于 a 举起电话机 然后 b 的号码 和 通话之后挂机的动作。

RTP 是媒体传输 相当于 电话通了 a 和 b 语音通话的动作。

SDP 是媒体描述 相当于 a 和 b 都使用普通话进行对话。

假如 a 和 b 没有一个共同 的语言 会导致协商不成功。就无法通话。比如 a 是中国人，b 是英国人。a 问：会说汉语吗。b 听不懂，b 问：can you speak English？a 也不懂。最终只能放弃通话。

SIP 是控制命令，是基于文本的，通过网络抓包是可以直接看出内容，这样调试排查问题就比较方便。SIP 协议 默认使用 UDP 协议。但是也支持 TCP 协议，有些变态的系统或者软电话只能支持 TCP，比如微软的 LYNC。

SIP 默认是使用 5060 端口。但是通常也可以指定其它端口。比如 5061 之类。有些软件，比如 tom 版本的 skype 启动之后也会使用 5060 端口，这样导致了其它软电话或者 FreeSwitch 启动失败。假如一个机器上要同时启动软电话和 FreeSwitch，就要注意它们使用的端口不能一样，否则必然会有一个启动不成功，或者表面看起来是启动成功，但是实际上不能用。

通常系统有很多线路可以并发使用，但是它们都是使用一个端口（默认是 5060）进行信令控制的。SIP 使用模式 有两种：一种是注册模式，大家每个人分配一个分机号码和密码，然后都注册到一个 SIP SERVER /IPPBX 上，呼叫的时候只要呼叫分机号码就可以。

另外一种是点对点模式： 通过呼叫 软电话的或者 IP 话机的 IP+端口（默认 5060）的模式。

标准的 sip 地址格式是： sip:56789@IP:PORT

其中 56789 是号码。

3. SIP 信令协议有哪些是必须掌握的？

偶认为 SIP 信令协议 以下这些是必须掌握的，不掌握这些基本东东，就算考试不及格，你要是搞不懂就跟乘法口诀一样，先背。

首先要记住的是消息类型：

消息	描述
INVITE	用于发起呼叫请求，包括消息头和数据区两部分。消息头包含主、被呼叫的地址，呼叫主题和呼叫优先级等信息。数据区则是关于会话媒体的信息，可由会话描述协议 SDP 来实现
BYE	结束 SIP 会话
OPTIONS	用于询问被叫端的能力信息，OPTIONS 本身并不能发起呼叫
ACK	对已收到的消息进行确认应答
REGISTER	用于用户向 SIP 服务器传送位置信息或地址信息
CANCEL	取消当前的请求，但它并不能中止已经建立的连接

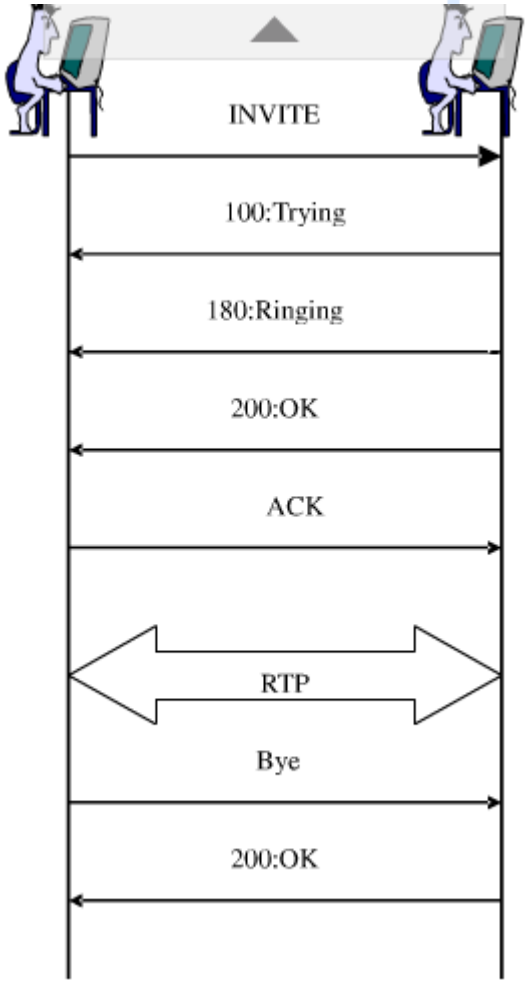
INVITE 表示开始呼叫，BYE 表示挂机，谁先发 BYE 就表示谁先挂机，这个会经常使用来查为啥掉线。

次要记住的是回复的代码类型：

代码类	描述	示例
1xx	表示收到了请求消息，正在处理中	180 尝试中
2xx	表示成功地接收到了消息，并理解和处理了请求	200 OK
3xx	表示刷新，需要进一步努力来完成请求	302 重定向
4xx	表示客户端出现了错误，如请求消息的语法错误或者不被服务器支持等	404 未发现
5xx	表示服务器端出现了错误	501 未执行
6xx	全局的错误，请求无法在任何服务器完成	603 退回

常用的返回代码是 200 OK 表示成功，480 ，530 之类的通常都是表示不行的。

最后要记住的是基本的 SIP 通信过程顺序图：

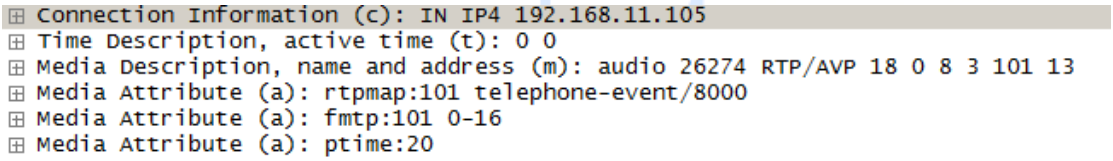


上面的三个图你要是都记住了，就算基本了解了 SIP 信令协议。

4. RTP 基本问题有哪些？

RTP 的默认端口是没有指定的，通常系统有很多线路可以并发使用，每一路通常都会占用两个端口 一个是 RTP 一个是 RTCP，
RTP 媒体传输的机器的 IP 和 SIP 的信令控制的 IP 可以不是同一个。虽然很多情况下它们是一个，表示信令控制和媒体传输都是一台机器上发起的。但是大的系统里面往往它们的 ip 不是同一个。小系统：一个部门就是一个人，这个人既是领导（SIP）也是干苦力活的（RTP）。
因为一个机器的 cpu 毕竟计算能力有限。而一个大的系统相当于一个部门很多人，领导（SIP）通常一个就够，干苦力活的（RTP）需要很多人。
SIP 的流量通常很少，但是 RTP 的流量通常是巨大的，因为要传输语音流个视频流。所谓：领导一句话，手下累死人。
通常的一次通话，SIP 一般是几个 UDP 数据包来回，但是 RTP 的 UDP 包通常是每秒 50 个。通话 2 分钟就是 6000 个数据包。

RTP 的语音流是一份一份（从时间上看是离散的）进行传输的。
一份就是一个 UDP 包，通常情况下一个包包括了 20ms 或 30 毫秒的语音。每个包的负载大小根据语音编码器不同，在几十 Byte 到几百 Byte 之间
这样的话，假如是 20ms 一个包，1 秒的语音要发 50 次，可见流量是很大的。
这个 RTP 的 ip 通过抓包可以看出来。
参考下图：



其中的 Connection Information 后面的 192.168.11.105 就是 RTP 的 IP 地址
audio 后面的 26274 就是 RTP 的端口。
可以这样理解 RTP：
语音就像我们的喝水一样理论上是连续的，但是你喝的时候总得一口一口喝吧，一个水就相当于一个 20ms 的 UDP 语音包，你不停地喝，几分钟之后就喝完了碗里的所有的水。

通常的一线 VOIP 通话，RTP 需要一个 UDP 端口，RTCP 也需要一个 UDP 端口(通常=RTP 的端口+1)。因此一线通话需要两个 UDP 端口。假如是 1000 线并发，就需要 2000 个 UDP 端口。比如从 10000 到 12000。

5. SDP 基本问题有哪些？

SDP 是对媒体的描述，描述了 RTP 的 IP 和端口 PORT
说明自己这边有哪些编码器（相当于一个人会说哪些语言）
也描述了是否支持视频（说白了就是有视频编码器）。
也描述了是否支持按键。
很多情况下呼叫建立不成功都是因为 SDP 协商不成功导致。这样需要通过抓包才能看出。
参考下图：

```
⊟ Session Description Protocol
    Session Description Protocol Version (v): 0
  ⊞ Owner/Creator, Session Id (o): 1003 1352630866 1352630886 IN IP4 192.168.11.105
    Session Name (s): Kapanga [1352630866]
  ⊞ Connection Information (c): IN IP4 192.168.11.105
  ⊞ Time Description, active time (t): 0 0
  ⊞ Media Description, name and address (m): audio 5100 RTP/AVP 0 8 101
  ⊞ Media Attribute (a): rtpmap:0 pcmu/8000
    Media Attribute (a): sendrecv
  ⊞ Media Attribute (a): silenceSupp:off - - -
  ⊞ Media Attribute (a): rtcp:5101
  ⊞ Media Attribute (a): maxptime:20
  ⊞ Media Attribute (a): ptime:20
  ⊞ Media Attribute (a): rtpmap:8 pcma/8000
  ⊞ Media Attribute (a): rtpmap:101 telephone-event/8000
  ⊞ Media Attribute (a): fmtp:101 0-15,36
```

其中的 Connection Information 后面的 192.168.11.105 就是 RTP 的 IP 地址
audio 后面的 5100 就是 RTP 的端口。
RTP/AVP 后面的 0 8 101 表示支持的 编码 0 表示 G.711 Ulaw, 8 表示 G.711 的 Alaw，101 是表示支持电话按键事件
后面的很多 Media Attribute 是对媒体属性的具体描述。
总结一下：SDP 的核心有几个：

A. 说明媒体的来源 IP 地址
B. 说明媒体的来源的端口 port
C. 说明自己有哪些媒体能力（能说几个语言）
D. 包括对 dtmf 按键的是否支持是说明。

一旦两个软电话要对话，就要对比交换上面这些信息。比如 FreeSwitch 默认要求 rfc2833 模式的 DTMF 按键(上面抓包里面的 rtpmap:101 telephone-event/8000)，但是假如软电话上没有设置 rfc2833 的 DTMF，那么 SDP 就没有 101，协商就不会成功，从而导致通话无法建立。

6. 常用的支持语音的软电话有哪些？

Windows 下常见的软电话有：
Xlite/ eyebeam,kapanga,linphone,jitsi，PJSIP 等等
Linux 下有 Linphone，PJSIP 等，安卓下有 Linphone，ImsDroid
安卓系统下常用的软电话有：sipdroid,CSipSimple,jPhoneLite
本人建议：初学者使用 Xlite/ eyebeam，linphone 测试，安卓下使用 Linphone，ImsDroid 测试。
为啥会有这么多，恐怕是因为很难找到一个完美的软电话能实现所有的功能，又适合所有的平台，又有源码可以二次开发。

7. 常用的支持视频的软电话有哪些？

Windows 和 Linux 下的软电话大多也支持视频。
Windows 下常见的软电话有：
Xlite/ eyebeam,kapanga,linphone,jitsi 等等
Windows 还有 pjsip 和基于 pjsip 的 MicroSip。
Linux 下有 Linphone 等
安卓下有 Linphone，ImsDroid 等
Windows 下本人使用 kapanga 和 Linphone 测试过 FreeSwitch 的视频通话。

也测试过 MicroSip 的视频通话，选择 H264 的编码，然后使用 p2p 模式的时候效果还是可以的。
安卓下本人使用两个 ImsDroid 测试过视频通话。
除非特别的地方，这些软电话的配置偶就不一一介绍。

8. Windows 下哪个开源的软电话比较好二次开发？

Windows 的软电话很多，开源的也不少，
从选用开源软件的角度：
1 要考虑项目的热度，假如一个开源项目很久没有更新了，不建议选择。
2 要考虑项目的完整性，假如一个项目七零八落的，没有网站的源码包，不建议选择。
3 要考虑项目的前瞻性，假如一个项目不够先进，不建议。

因此我建议使用 pjsip。遗憾的是编译版本 pjsip 默认没有视频支持，要支持视频很麻烦，需要 QT 等 7788 的。
因此就有了另外一个 MicroSip，是基于 pjsjp 编译的 lib 上编译的有 UI 的而且有源码。
下载地址：
http://www.microsip.org/downloads

9. Windows 下开源 PJSIP 软电话如何编译？

PJSIP
项目地址：http://www.pjsip.org/
源码下载地址：http://www.pjsip.org/download.htm
假如要支持视频，需要下载 2.X 系列源码包，2.X 系列需要使用 VS2008 编译。

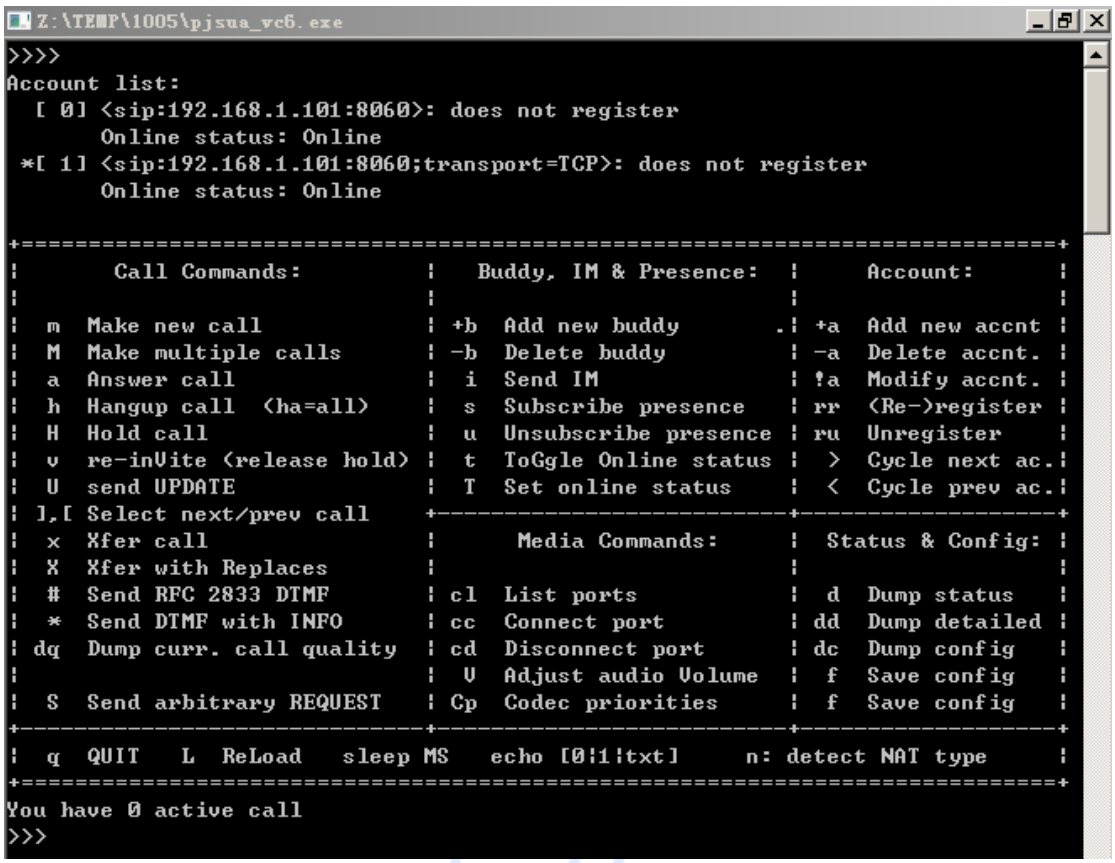
假如只有 VC6 编译器，可以选择 1.X 系列的源码包，比如 1.16，1.X 不支持视频。
以 1.16 版本为例，下载之后解压。然后构造 config_site.h 内容如下：

```
#define PJSUA_MAX_CALLS                 256   //最大通话数
#define PJ_LOG_MAX_LEVEL      1   //日志级别，级别越高日志越详细
#define PJMEDIA_HAS_L16_CODEC           0   //禁用 L16
#define PJMEDIA_HAS_G722_CODEC          0   //禁用 722
#define PJMEDIA_HAS_SPEEX_CODEC         0   //禁用 SPEEX
#define PJSIP_MAX_TRANSPORTS      512       //最大通话的 2 倍
#define PJ_IOQUEUE_MAX_HANDLES          512   //最大通话的 2 倍
```

放到
pjproject-1.16\pjlib\include\pj 目录下。
最后打开里面的 dsw 直接编译 就可以，简单明了。

以 1.16 版本为例，默认编译之后，会 在 pjproject-1.16\pjsip-apps\bin
目录下产生 pjsua_vc6d.exe 文件，

Pjsua 是控制台的程序，可以很容易集成到 GUI 的 windows 程序里面。
也可以很容易用它来做被叫，来做 FreeSwitch 的压力测试。
稍微修改一下，也可以作为主叫来做 FreeSwitch 的压力测试。
后续性能测试部分会专门讨论这个事情。
pjsua_vc6d.exe 可以直接运行，默认没有参数的运行是 p2p 的模式。
运行界面如下：



界面上提供了很多命令，比如可以 Cp 来看编码器的属性，直接回车就可以看到当前有多少 active call，等等。

Pjsua 提供了巨多的参数输入选择：
下面列举几个：
注册模式运行，分机 1002， sip ip 是 192.168.1.101，密码是 1234：
pjsua_vc6d.exe --id sip:1002@192.168.1.101 --registrar sip:192.168.1.101
--realm 192.168.1.101 --username 1002 --password 1234 --reg-timeout 300
其它常用参数：
自动应答：
--auto-answer 200
允许最大呼叫个数
--max-calls 4
允许最大呼叫个数
--thread-cnt 4
允许最大通话时间，单位秒
--duration=90

不使用 tcp，仅仅使用 udp 作为 sip 的传输
--no-tcp
没有声卡
--null-audio
通话之后，自动播放语音
--auto-play --play-file 1.wav
指定本地 sip 的端口
--local-port=5100
指定本地 rtp 的开始端口
--rtp-port=5110
自动回路，假如它作为被叫，这样你拨打它，喊话就会听到自己的声音。
--auto-loop

10. Android 下哪个开源的软电话比较好用又好上手？

Android os 下的软电话很多，开源的也不少。Linphone 是一个不错的选择，遗憾的是在安卓下编译使用 linphone 对一般的伙计是一个艰巨的任务。

在本文写的时候 PJSIP 还未原生发布 Android 的版本，不过有人在它的基础上包装了 Android 的版本 叫做 CSipSimple。

根据上面说的选用开源软件的 3 个角度。也不建议使用这个，因为不是原生支持的。等 PJSIP 原生的发布支持了，我会补充评估。

建议选择 ImsDroid 开源的软电话

11. Android 下 ImsDroid 开源软电话如何编译？

ImsDroid
项目地址 ： http://code.google.com/p/ImsDroid/
源码下载。使用 svn ：
svn checkout http://ImsDroid.googlecode.com/svn ImsDroid
取出之后的目录如下:
ImsDroid\branches\2.0 目录就是源码
子目录如下：

android-ngn-stack	2013/6/7 17:30	文件夹	
apk-debug	2013/6/7 14:44	文件夹	
images	2013/6/7 14:45	文件夹	
imsdroid	2013/6/7 19:57	文件夹	
myFirstApp	2013/6/7 19:39	文件夹	
native-debug	2013/6/7 14:45	文件夹	
testCall	2013/6/7 14:48	文件夹	
testRegistration	2013/6/7 14:48	文件夹	
tests-apk	2013/6/7 14:48	文件夹	
testT140	2013/6/7 14:45	文件夹	
android-ngn-stack-00.pdf	2013/6/7 14:48	Adobe Acrobat ...	377 KB

其中：

ImsDroid\branches\2.0\android-ngn-stack 是 ngn 的协议栈

ImsDroid\branches\2.0\ImsDroid 是安卓的程序。

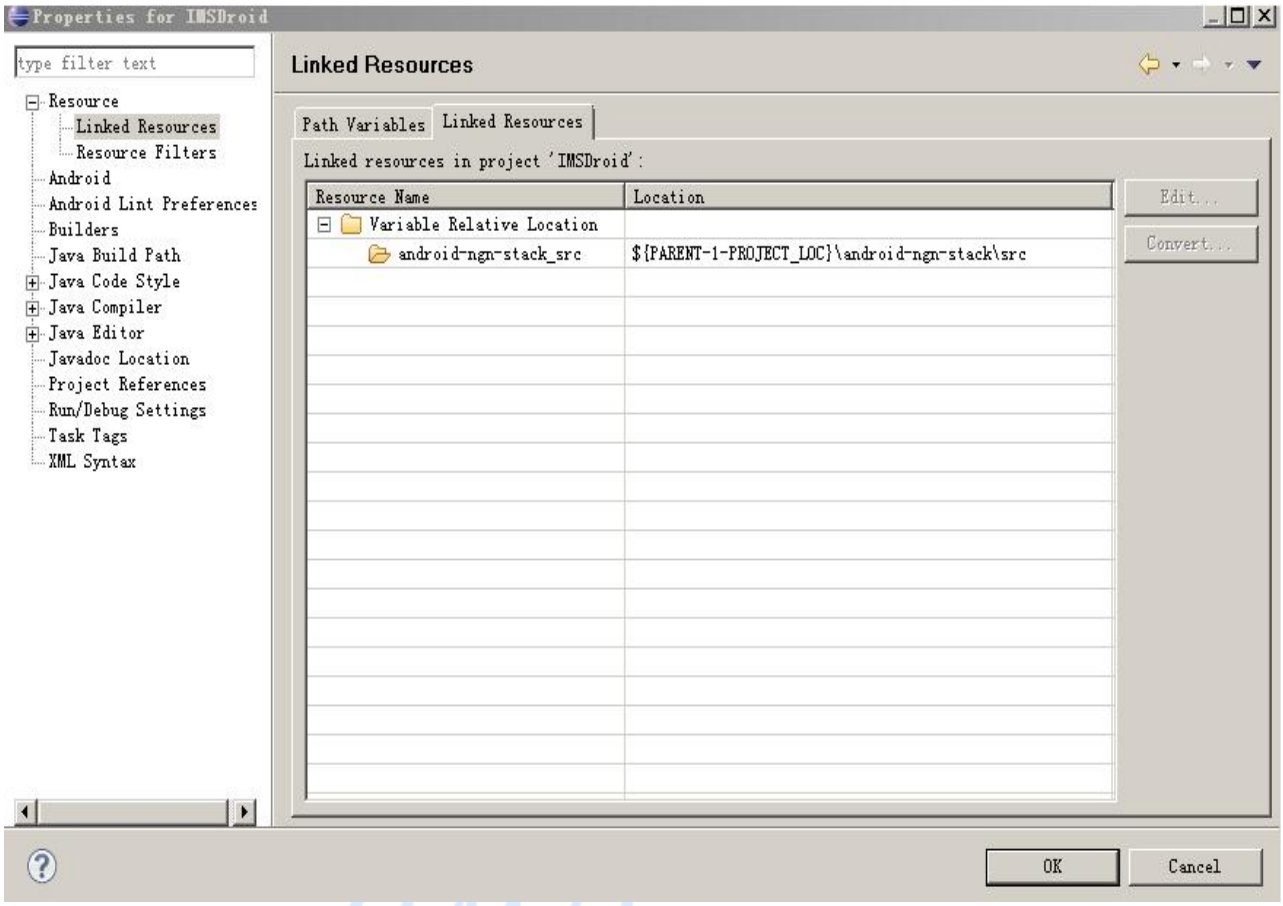
使用 eclipse 打开 ImsDroid\branches\2.0\android-ngn-stack 工程

和 ImsDroid\branches\2.0\ImsDroid 工程

编译。设置安卓的 SDK 版本为最新的版本。因为 ImsDroid\branches\2.0\ImsDroid 工程 使用 ImsDroid\branches\2.0\android-ngn-stack 工程作为资源库文件。因此需要

然后设置 ImsDroid\branches\2.0\ImsDroid 工程里面的 linked Resources 里面的 android-ngn-stack-src 的目录为合适的 ImsDroid\branches\2.0\android-ngn-stack\src

如下图：



编译通过了说明源码没有问题，否则有问题的地方手工修改一下源码。

我使用 4.2.2 的 sdk 编译，NgnSipStack.java 会有错误,注释掉，或者使用固定字符串代替，修改地方如下：

。。

//import org.doubango.ngn.R;//yhy2013-06-07

。。。

super.addHeader("User-Agent", String.format("IM-client/OMA1.0 android-ngn-stack/v%s (doubango r%s - %s)",
NgnApplication.getVersionName(),
//NgnApplication.getContext().getString(R.string.doubango_revision), //yhy2013-06-07
"000", Build.MODEL));

。。。

编译通过了。需要说明的是 ImsDroid\branches\2.0\ImsDroid 工程无法直接导出 apk 的文件。

因此需要自己手工建立一个空 android 的工程。比如叫做 ImsDroid2，工程目录与 ImsDroid 目录同级，比如：ImsDroid2.0

然后从 ImsDroid 目录下拷贝

AndroidManifest.xml

src 目录

res 目录

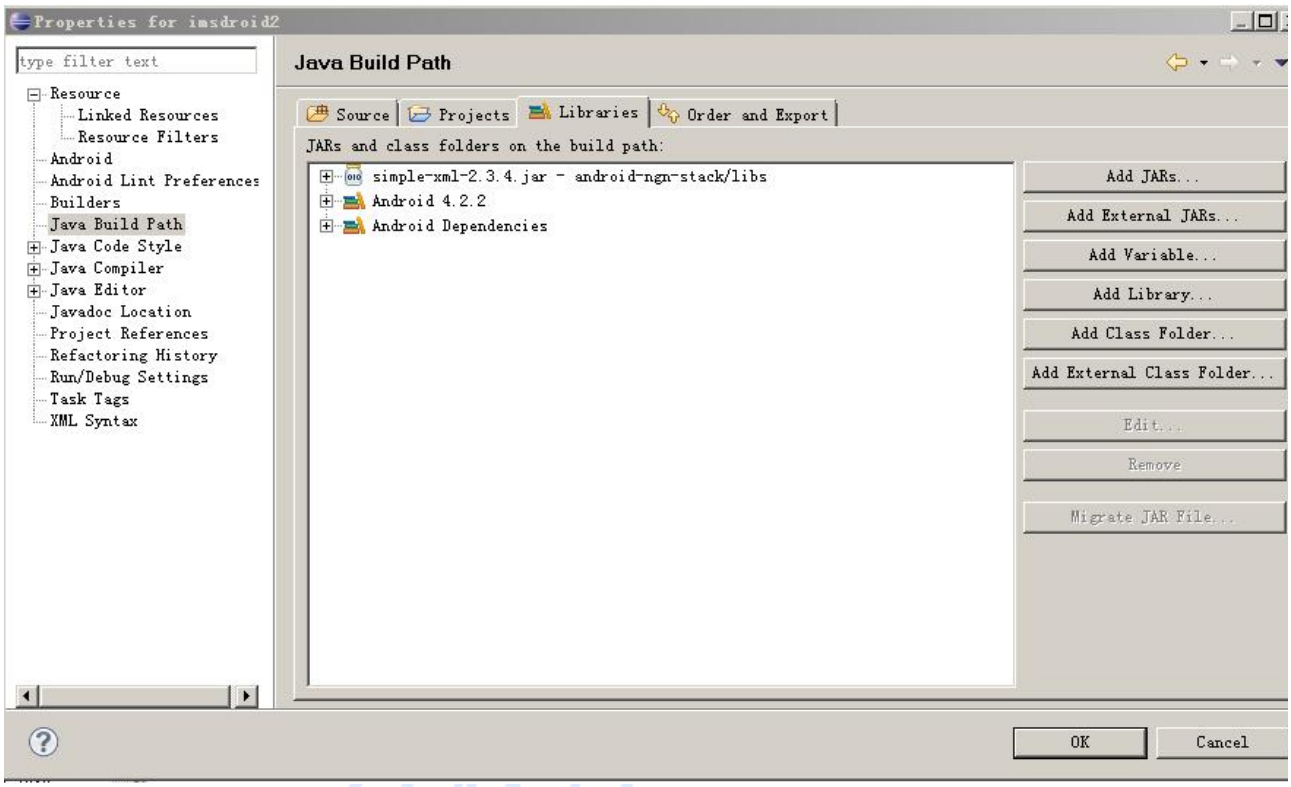
gen 目录

从 android-ngn-stack 目录下 拷贝

libs 目录

然后设置 ImsDroid2 工程里面的 linked Resources 里面的 android-ngn-stack-src 的目录为合适的 ImsDroid\branches\2.0\android-ngn-stack\src 目录

然后添加设置 android-ngn-stack.jar 库 如下图：



然后 编译，最后打包成 apk 文件就可以使用到安卓手机上了

12. 常见语音编码器有哪些？

常用的语音编码器有

G.711 Ulaw 和 Alaw 这个是绝大多数的 VOIP 都支持的。G.711 要是都不支持，就不能称为 VOIP，所以刚刚开始测试使用某个系统或者软电话，使用 G.711 准没有错。

我建议最开始测试的时候就只选择 alaw（或者 ulaw 也行）语音编码一种。

我喜欢选择 alaw 是因为之前搞过语音处理，alaw 的完全静音是 0xD5 或者 0x55,

如下图：

```
00000320h: D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 照照照照照照照照
00000330h: D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 55 D5 D5 ; 照照照照照照照証照
00000340h: D5 D5 55 D5 D5 D5 D5 D5 D5 55 D5 D5 55 D5 D5 D5 ; 照U照照照照U照U照?
00000350h: 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 55 ; U照照照照照照照照U
00000360h: D5 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 証照照照照照照照照
00000370h: D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 照照照照照照照照照
00000380h: D5 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 証照照照照照照照照
00000390h: D5 D5 D5 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 55 D5 ; 照証照照照照照照U?
000003a0h: D5 D5 D5 D5 D5 D5 D5 D5 D5 55 D5 D5 D5 D5 55 D5 ; 照照照照証照照U?
000003b0h: D5 D5 D5 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 照証照照照照照照照
000003c0h: D5 55 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 D5 ; 証照照照照照照照照
```

录音的语音文件使用 UltraEdite 直接打开然后选择 16 进制，使用眼睛看二进制大概就能看出是否有问题。否则就需要 CoolEdite 之类的语音播放软件进行播放才能知道是否语音有问题。

G.729 这个是公网上使用最多的，但是由于有授权，有些软电话不支持，或者需要采购才能使用。

iLBC 大多数软电话都支持，而且没有授权限制
SPEEX 大多数软电话都支持，网络上也有源码，siri 和科大的语音识别的的语音传输就是使用这个编码，flash player 也支持这个编码
GSM 大多数软电话都支持，在公网上假如 G.711 带宽太大，其它编码又没有的情况，可以考虑使用这个。
SILK 这个是 Skype 刚刚开源的编码器，有 8K 16K 各种采样，pjsip 和 linphone 支持。
Opus 是刚刚问世不久的开源免专利费音频编解码器 是 2012 Opus 正式成为 IETF 标准（RFC6716）

13. 常见视频编码器有哪些？

H.263 最基本的视频，大多数的支持视频的软电话都支持
其中 H.263 又细分为
H.263
H.263+/ H.263-1998
H263++/H.263-2000 等
MP4
H.264 (MPEG-4 part 10) H.264 根据 profile 不同可以分为：
H.264 (BP)
H.264 (MP)
VP8 这个是 google 开源的最新的编码，基于 H.264 改进的编码器，号称效果最好（很多人认为是吹牛 B）。

14. PSTN 和 VOIP 区别有哪些？

PSTN 是电路交换，是时分复用，话音质量是可靠的可控的，无论闲死还是忙死，话音质量都是一样的。无论电话是否在用，某个时间片内的传输的内容一定是某个时隙的（或者说某线的）。
VOIP 是数据包交换，话音质量有用以太网的天生属性，质量是不可靠的。通常网络情况下话音效果不错，但是一旦网络繁忙，话音质量就立刻下降。因此 VOIP 的运营最大的难点是在于系统的稳定性。

15. PSTN 常用信令有哪些

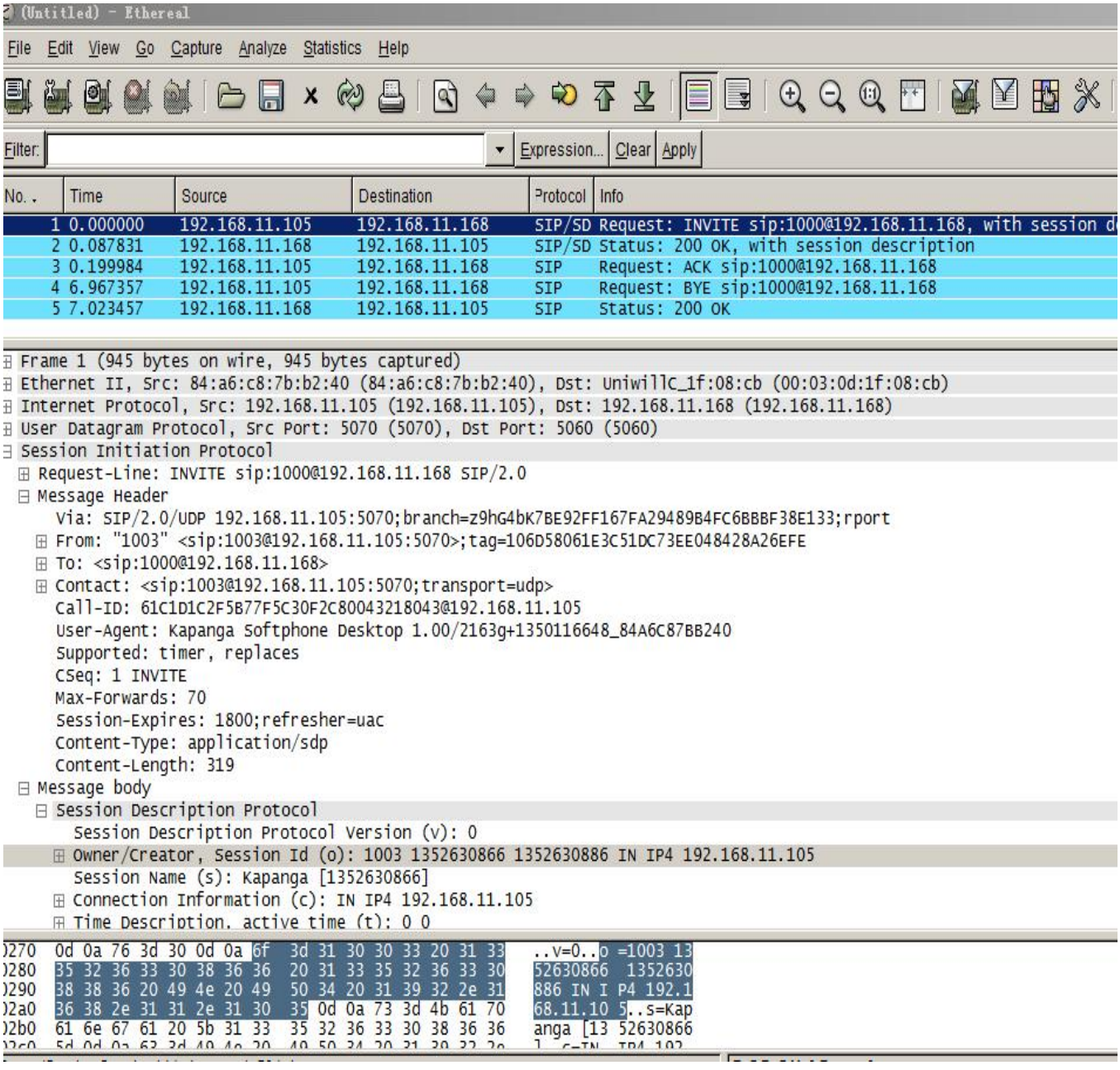
常用的 PSTN 信令有 ISDN PRI 和 SS7
其中区别在于 ISDN PRI 是一个 E1（32 个时隙，30 路话路）使用其中的 16 时隙作为信令控制
SS7 是 7 号信令，是很多 E1，最大可以到 2^12 路（4096）话路共用某个 E1 的某个时隙（通常是 16 时隙，作为信令 LINK）作为信令控制，为了提高可靠性，通常使用两个 E1 线路上的时隙（即两个 LINK）作为信令控制。
这个跟 Sip 特别像，之前提到过 sip 只要一个端口就可以，但是 RTP 可以是几千端口。
比如 1000 线的并发，sip 只要 5060，RTP 需要 10000-12000 总共是 2000 个 UDP 端口。

16. VOIP 的系统开发和测试有哪些常用工具？

最常用的通常是网络抓包程序

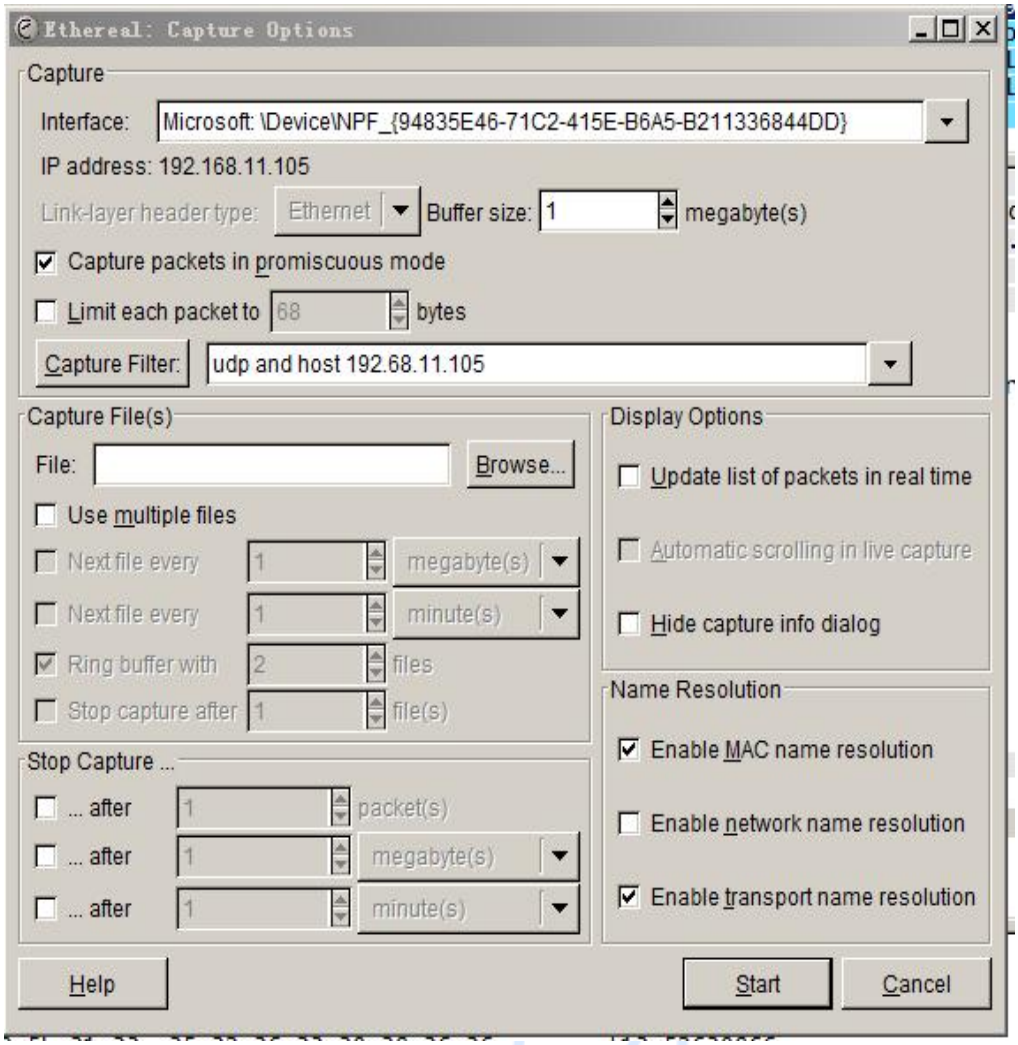
Ethereal + WinPcap　或者　wireshark + WinPcap

有用 sip 协议是文本字节流模式，抓包之后，直接使用工具里面的查看窗口直接查看，以 Ethereal 为例如下图：

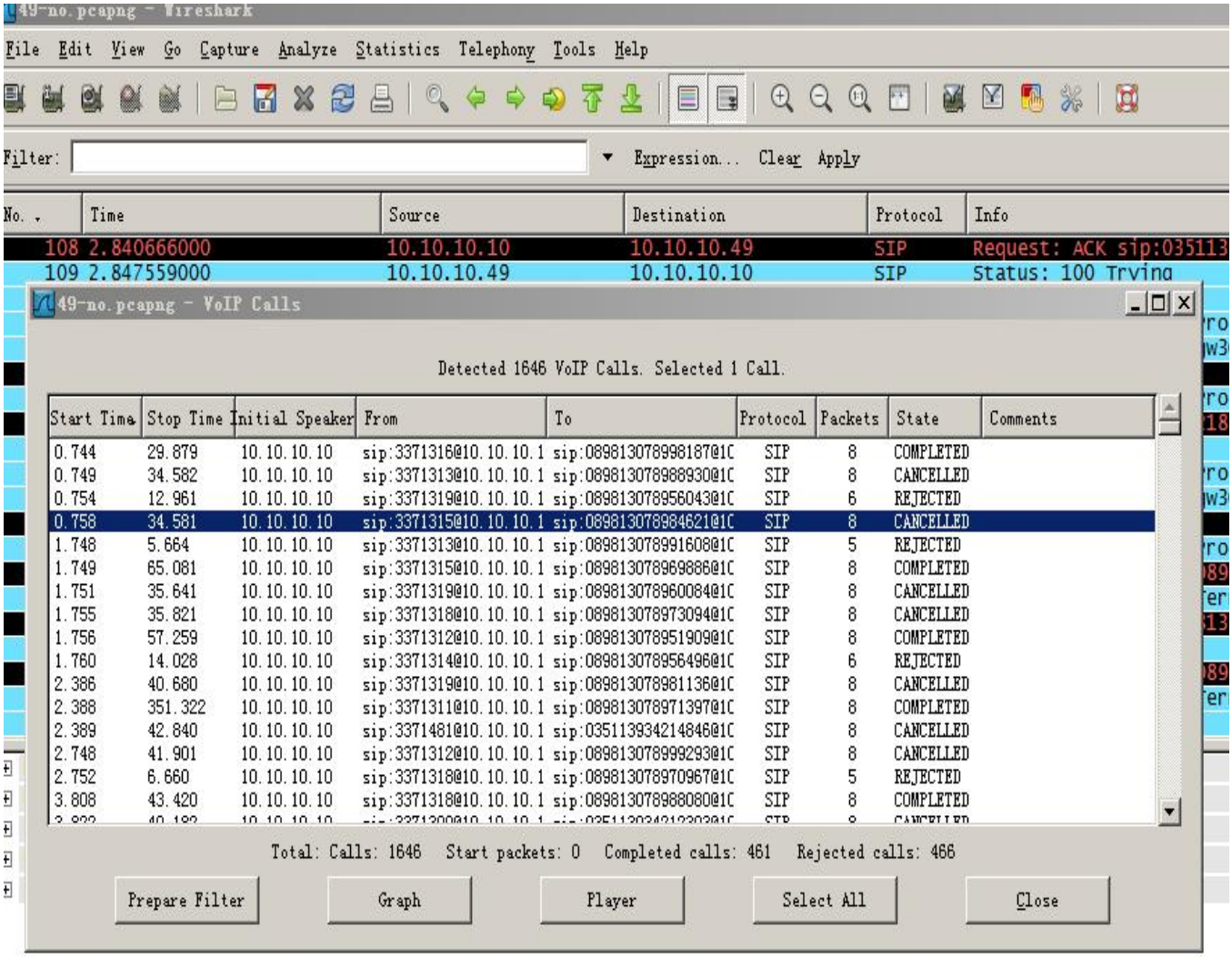


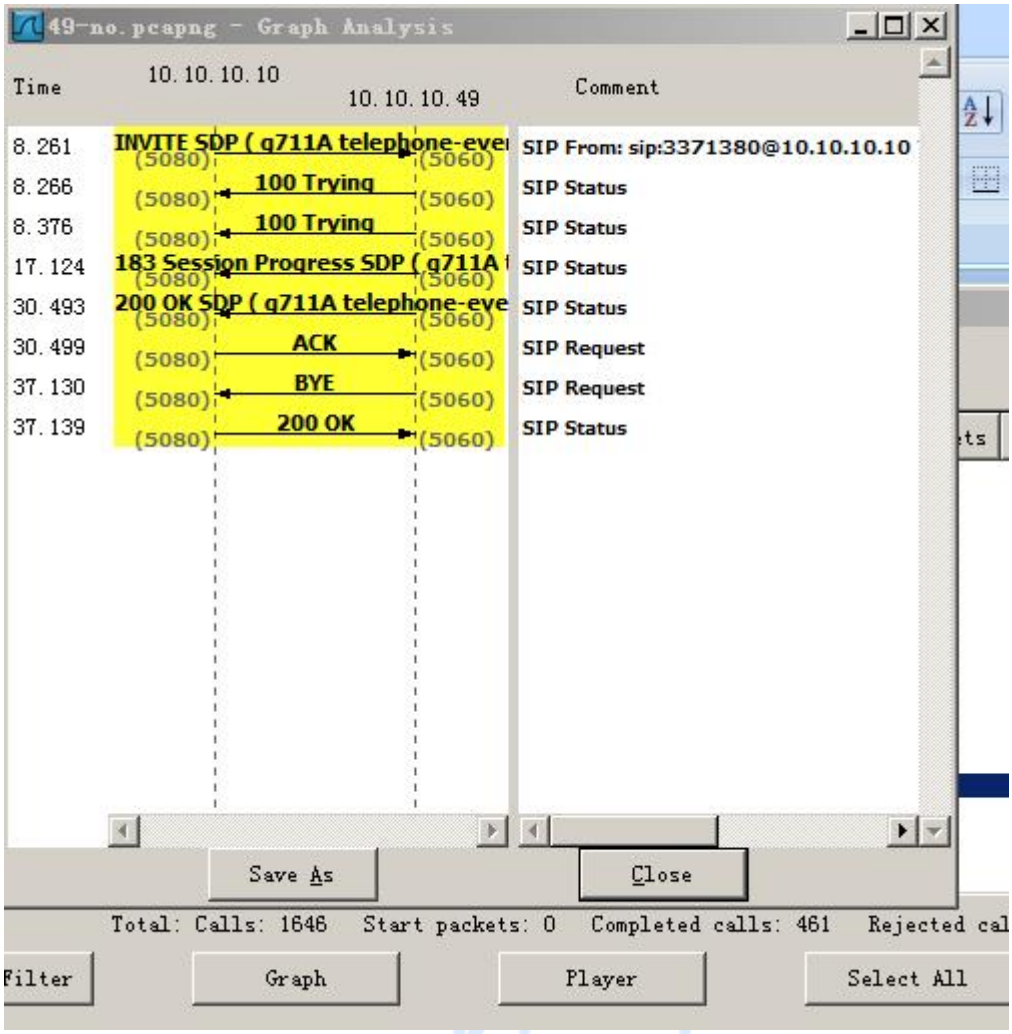
17. 如何使用 Ethereal，Wireshark 对指定机器进行抓包分析？

在 Capture 菜单中选择 Option 子菜单。然后在出现的对话框中选择网卡（Interface）
如下图：



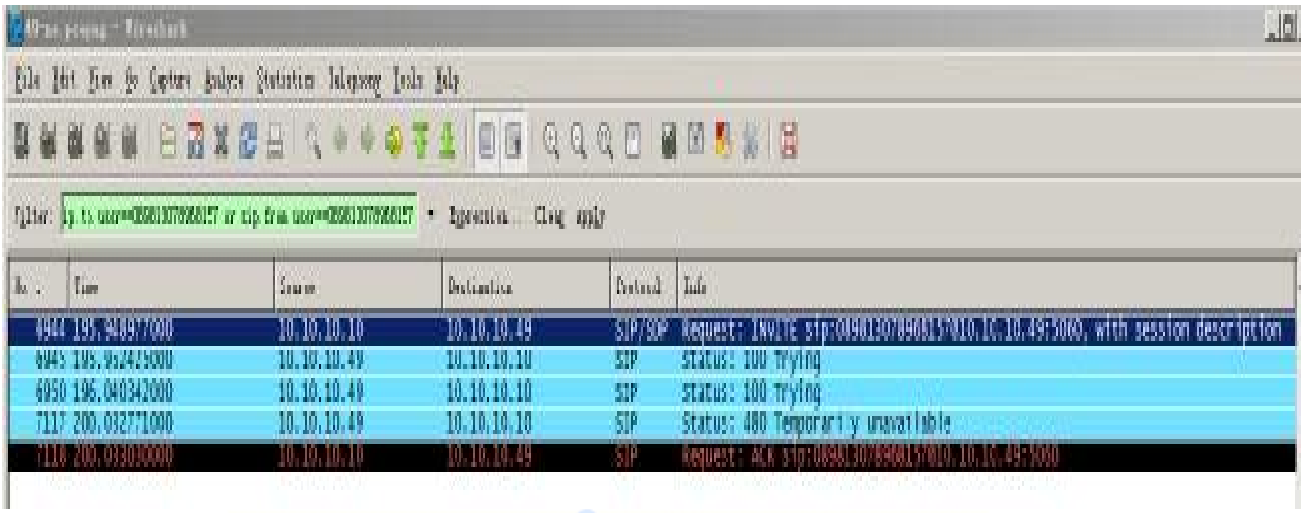
然后在 Capture Filter 里面输入 udp and host 192.168.11.105
Udp 表示 只是抓 udp 的包，因为 sip 默认是基于 udp，rtp 也是基于 udp
Host 192.168.11.105 表示只抓 ip 是 192.168.11.105 的数据包（从本机器发给 192.168.11.105，或者从 192.168.11.105 发给本机）。
其中 192.168.11.105 也可以是本机地址。
然后点 start 开始抓包。
也可以使用 Wireshark 对指定的机器抓包，条件是一样的，Wireshark 好处是
可以使用 Telephone 菜单 下的 VoIP Calls 菜单对抓包数据进行分析，可以对选择的某个会话点 Gragh 图形显示 sip 的会话过程，假如也有抓媒体的话，甚至可以点 Player 进行语音播放。这个在具体要分析通话媒体的时候比较有用如下图：


49-no.pcapng - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Info
108 2.840666000 10.10.10.10 10.10.10.49 SIP Request: ACK sip:035113
109 2.847559000 10.10.10.49 10.10.10.10 SIP Status: 100 Trying
49-no.pcapng - VoIP Calls
Detected 1646 VoIP Calls. Selected 1 Call.
Start Time Stop Time Initial Speaker From To Protocol Packets State Comments
0.744 29.879 10.10.10.10 sip:3371316@10.10.10.1 sip:089813078998187@1C SIP 8 COMPLETED
0.749 34.582 10.10.10.10 sip:3371313@10.10.10.1 sip:089813078988930@1C SIP 8 CANCELLED
0.754 12.961 10.10.10.10 sip:3371319@10.10.10.1 sip:089813078956043@1C SIP 6 REJECTED
0.758 34.581 10.10.10.10 sip:3371315@10.10.10.1 sip:089813078984621@1C SIP 8 CANCELLED
1.748 5.664 10.10.10.10 sip:3371313@10.10.10.1 sip:089813078991608@1C SIP 5 REJECTED
1.749 65.081 10.10.10.10 sip:3371315@10.10.10.1 sip:089813078969886@1C SIP 8 COMPLETED
1.751 35.641 10.10.10.10 sip:3371319@10.10.10.1 sip:089813078960084@1C SIP 8 CANCELLED
1.755 35.821 10.10.10.10 sip:3371318@10.10.10.1 sip:089813078973094@1C SIP 8 CANCELLED
1.756 57.259 10.10.10.10 sip:3371312@10.10.10.1 sip:089813078951909@1C SIP 8 COMPLETED
1.760 14.028 10.10.10.10 sip:3371314@10.10.10.1 sip:089813078956496@1C SIP 6 REJECTED
2.386 40.680 10.10.10.10 sip:3371319@10.10.10.1 sip:089813078981136@1C SIP 8 CANCELLED
2.388 351.322 10.10.10.10 sip:3371311@10.10.10.1 sip:089813078971397@1C SIP 8 COMPLETED
2.389 42.840 10.10.10.10 sip:3371481@10.10.10.1 sip:03511393421484 6@1C SIP 8 CANCELLED
2.748 41.901 10.10.10.10 sip:3371312@10.10.10.1 sip:089813078999293@1C SIP 8 CANCELLED
2.752 6.660 10.10.10.10 sip:3371318@10.10.10.1 sip:089813078970967@1C SIP 5 REJECTED
3.808 43.420 10.10.10.10 sip:3371318@10.10.10.1 sip:089813078988080@1C SIP 8 COMPLETED
Total: Calls: 1646 Start packets: 0 Completed calls: 461 Rejected calls: 466
Prepare Filter
Graph
Player
Select All
Close



18. 如何使用 Ethereal，Wireshark 对指定端口进行抓包分析？

类似上面问题，
在 Capture Filter 里面输入 udp and port 5060
Udp 表示 只是抓 udp 的包
Port 5060 表示只是抓 5060 端口上的包。
因为 sip 默认是基于 udp，默认是 5060 端口。以上的命令就是只是抓 sip 信令控制的包。
对抓出来的包通常非常多，可以使用过滤条件进行过滤，以 Wireshark 为例，
我们要查 通过网关呼叫到 089813078968157 号码的情况，在 Filter 里面输入，
sip.to.user==089813078968157 or sip.from.user==089813078968157
然后点 Apply



发现不通。

19. Ethereal 能对本机内的通信进行抓包吗?

本机的内地通信实际上没有经过网卡，是内部 LoopBack。
因此 Ethereal 无法实现对本机内的通信进行抓包，
比如本机的软电话呼叫本机上的 FreeSwitch，就无法抓到任何包。
要想抓包只能是经过网卡，进出口的包，相当于海关上才能查是否走私一样。

20. Ethereal 能对其它机器之间的通信进行抓包吗?

一般情况下，目前网络的基础设施都是网络交换机假如 A 机器和 B 机器进行通信，而你要在 C 机器上对 A 机和 B 机的通信进行抓包，是抓不到包的
比如 a 机的软电话呼叫 B 本机上的软电话，你要在 C 机器上对它们的通信进行抓包，是抓不到包。
要实现对其它机器抓包，需要使用抓包的机器和被抓包的机器都连接到 HUB，这样就可以对其它机器进行抓包。

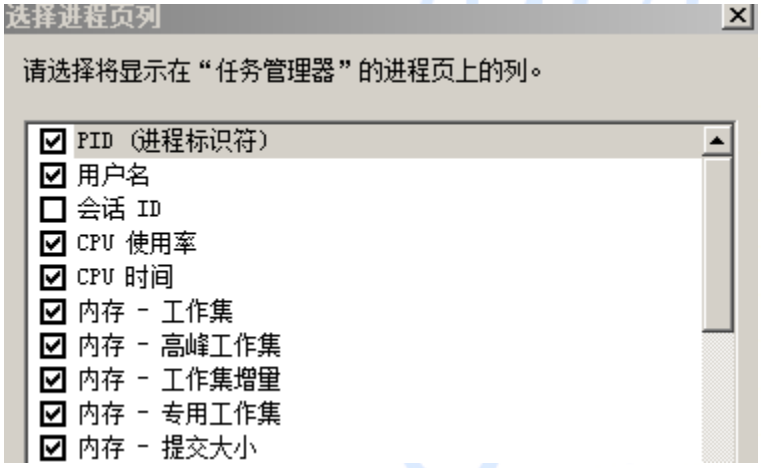
21. Windows 下使用啥命令工具可看哪个 port 被谁占用？

首先：在 windows 命令行下：输入
netstat -oan >c:\tmp.txt

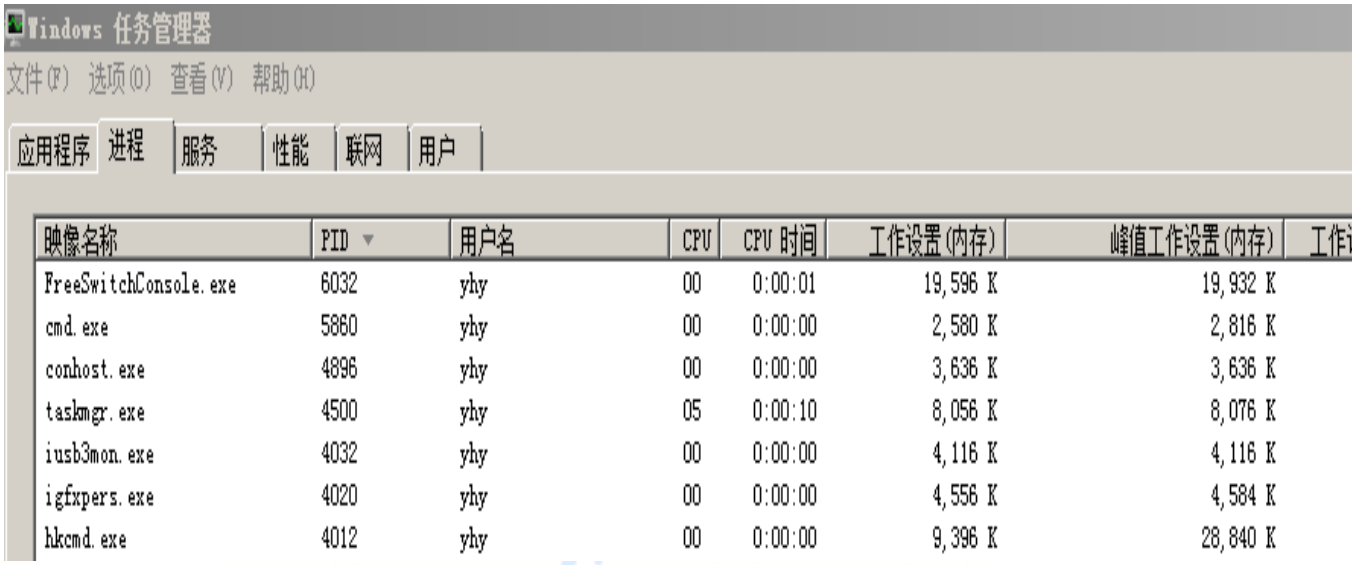
然后打开 c:\tmp.txt,如下图：

协议	本地地址	外部地址	状态	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	844
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING	484
TCP	0.0.0.0:1026	0.0.0.0:0	LISTENING	928
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING	996
TCP	0.0.0.0:1028	0.0.0.0:0	LISTENING	576
TCP	0.0.0.0:1029	0.0.0.0:0	LISTENING	600
TCP	127.0.0.1:2735	127.0.0.1:2736	ESTABLISHED	3444
TCP	127.0.0.1:2736	127.0.0.1:2735	ESTABLISHED	3444
TCP	127.0.0.1:8021	0.0.0.0:0	LISTENING	6032
TCP	192.168.11.105:139	0.0.0.0:0	LISTENING	4
TCP	192.168.11.105:5060	0.0.0.0:0	LISTENING	6032
TCP	192.168.11.105:5080	0.0.0.0:0	LISTENING	6032
UDP	127.0.0.1:60908	*:*		3244
UDP	127.0.0.1:62781	*:*		6032
UDP	127.0.0.1:62782	*:*		6032
UDP	127.0.0.1:62783	*:*		6032
UDP	127.0.0.1:62784	*:*		6032
UDP	127.0.0.1:62785	*:*		6032
UDP	127.0.0.1:62786	*:*		6032
UDP	192.168.11.105:137	*:*		4
UDP	192.168.11.105:138	*:*		4
UDP	192.168.11.105:1900	*:*		3244
UDP	192.168.11.105:5060	*:*		6032
UDP	192.168.11.105:5080	*:*		6032

可以看到 UDP 5060 端口的占用的 PID 是 6032。
从图中我们看到 6032 还占用了 5080 等端口。
打开任务管理器，点查看菜单，然后点选择列。。。在出现的对话框中勾选 PID，如下图：



然后在进程标签页总找出 PID 是 6032 的进程名称。如下图:



从图中可以看出是 FreeSwitchConsole.exe。我们知道这个是 FreeSwitch 的启动进程。

22. 如何根据使用的编码器计算 VOIP 需要的带宽？

VOIP 的数据包 包括 SIP 数据包和 RTP 数据包，SIP 只是传输控制命令，跟 RTP 的语音媒体数据包比较起来简直就是九牛一毛，完全可以忽略不计。

下面仅仅计算 RTP 的数据包

RTP 的数据包=以太网地址+IP 类型+IP 包头+UDP 包头+RTP 包头+语音编码之后的负载

以太网地址 12 字节

IP 类型 2 字节

IP 数据包头:20 字节

UDP 协议的报头: 8 个字节

RTP 包头是 12 字节 这些合计是 54B 大小是固定开销，只要使用 RTP，无论啥编码器都是必须的，以大多数的编码器每个包 20ms，每秒 50 个 RTP 包计算，这个开销每秒是 54*50*8=20.8kbps 是固定的开销。再加上语音编码的负载就是总开销。

以 G.711 alaw 编码器为例

语音负载：160 字节：一个包 20ms，8K 的采样频率，每个采样点使用 alaw 编码之后是 1 个字节。因此 20ms 的语音在编码之后的负载是 160 字节。

总共是 12+2+20+8+12+160=214 字节

1 秒的语音就是 214*（1000/20）=214*50=10700 字节,按照流量计算就是 10700*8/1024=84Kpbs 左右。

以 G.729 编码器为例:

假如是 G.729. 一个包 20ms 其它不变，只是语音编码之后变成了 20 字节，因此语音负载是 20 字节，可以算出来：总共是 12+2+20+8+12+20=74 字节

语音就是 74*（1000/20）=74*50=3700 字节,按照流量计算就是 3700*8/1024=29Kpbs 左右。

假设你有一个 10Mbps 的出口带宽，那只是理论带宽，做 VOIP 要考虑丢包，在不考虑启用 VAD 等节省带宽的因素情况下，根据本人测试结果，通常带宽实际使用按照 40-50%考虑有效带宽,也就是说带宽 4-5M 左右 VOIP 丢包比较小，通话不太会受影响，可以接受。

以 G.711 alaw 为例子:一线需要的带宽是 84Kbps.因此可以支持的线路数是：5*1000/84=60 线左右。
但是考虑到业务通常还有带宽需求，比如 web 弹屏和 web 页面啥的，因此，30 线是比较保险的。
以 G.729 为例子:一线需要的带宽是 24Kbps.因此可以支持的线路数是：5*1000/24=200 线左右。
但是考虑到业务通常还有带宽需求，比如 CRM 的弹屏通常包括了用户 360 度的信息和历史记录，数据量也是不小。因此，10M 的带宽，100 线以内是比较能保证话音质量的。

另外要注意的是 ADSL 的带宽上行和下行是不对称的，而 VOIP 的语音通话的开销是求上下行一样的带宽。因此假如并发使用的线路多了，而且碰到单边效果不好的情况，很可能的带宽不对称导致的。

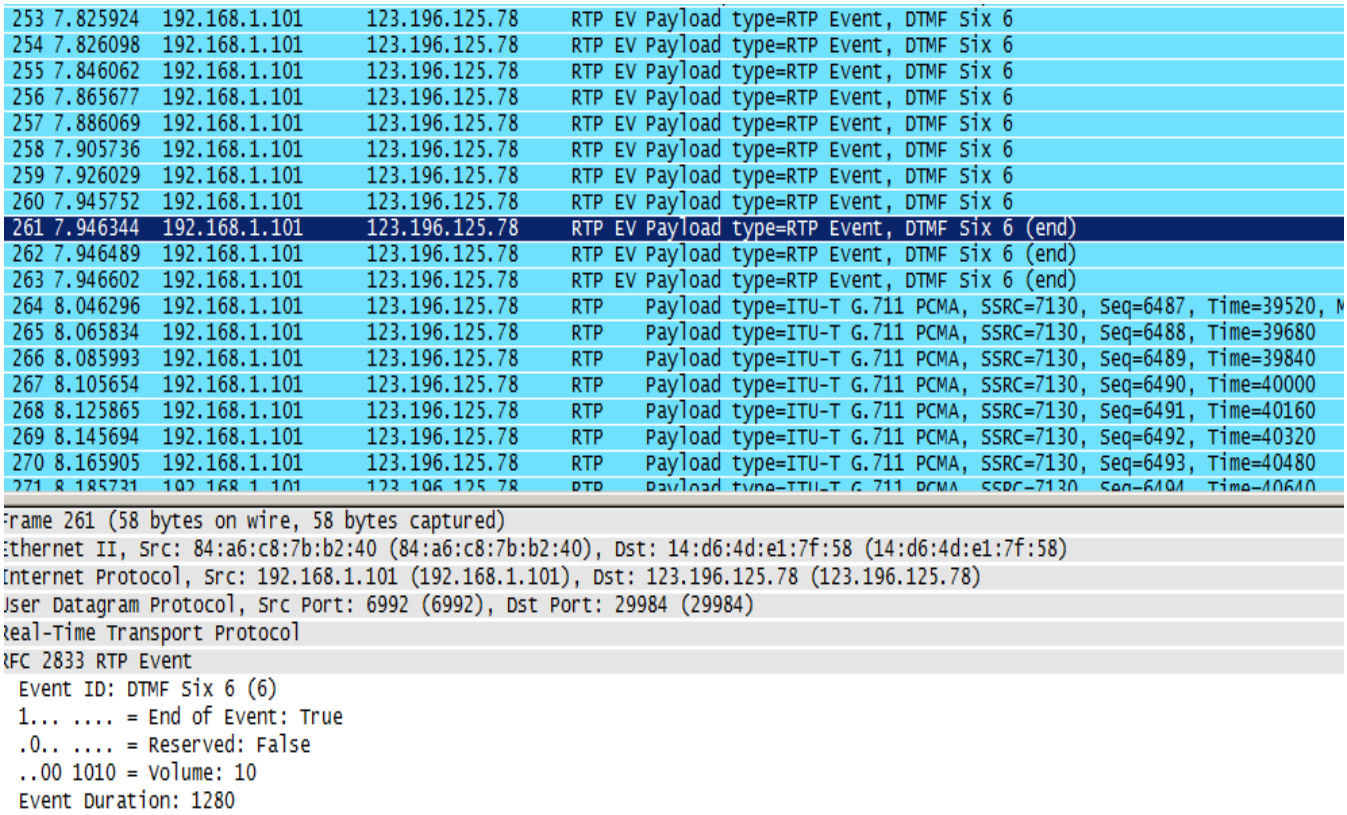
23. 如何测试你的系统的 WAN 的进出口带宽？

有很多第三方的网站都可以提供带宽测试
http://www.speedtest.net/
是比较好用的一个
上面有很多 host 可以选择进行测试。
比如你是联通的宽带，可以选择上面的联通的 host 进行测试。
根据本人测试的结果还是比较符合实际情况。

24. 如何理解 VOIP 里面的 DTMF 按键？

VOIP 里面的 dtmf 按键传输模式和 PSTN 里面的 DTMF 按键传输模式不大一样，
PSTN 环境一下 DTMF 按键是跟随语音流，一起传输模式的，可以理解为 IN BAND（带内）
VOIP 里面的 dtmf 基本上有 3 种传输模式：

A. 跟 pstn 一样的带内 DTMF（inband），这个传输模式在使用 G.711 编码的时候是可以的。在使用其它高压缩编码的时候不建议采用这种。
B. 按照 rfc2833 格式编码的 DTMF 按键传输模式，这种模式 下 DTMF 也是跟随 rtp 流一起的，但是又不大一样，有自己的特殊编码，大约 10-12 个包左右。如下图 表示 dtmf 按键的 6：



目前，这种情况使用的比较多。缺点，按键之后大多数的软电话或者系统会导致数秒的语音停顿，比如 voip 的 ivr 提示用户：按键开始录音，用户按#，然后开始说：1，2，3，4,5,6。。。

系统录制下来的语音可能 1，2，3 都丢了，只有 4,5,6...

C. 在 sip 信令里面传输的 sip-info 模式。不跟媒体 rtp 语音流一起走。而是在 sip 消息上进行传输。

这个比较高级，有些软电话或者系统不支持。

上面 3 种 FS 都支持，甚至可以同时支持里面的几种。比如同时支持 2833 和 sip info

25. 如何确认 PSTN 被 IP 化？

上面说过 PSTN 环境下 DTMF 按键是跟语音一起走的。但是现在很多 PSTN 无论是长途还是本地跨运营商，或者运营商内部，都 IP 化（走 VOIP）了，如何确认 PSTN 是否被 IP 化?

偶是这样确认的：通过按键+录音可以确认。

搞一个 ivr 系统，然后对拨入的电话语音提示：

“请按键之后开始录音”，假如用户按键打断播音，然后开始录音。

那么假如没有 IP 化，这个录音一般是正常的，但是假如 IP 化，

按键之后通常 5 秒内的说话录音是录不到的。5 秒后就正常。

归纳起来就是： 主叫按键之后假如被叫几秒内听不见主叫的声音，就是走 VOIP 了。

或者通过电话直接拨打，一个按键然后马上吼 12345。。。对方假如听不到开始的 123 之类就可以确认是走 IP 了。话音走 IP 对一般的通话没有啥影响，但是对精确度要求很高的某些需求就会影响，比方会影响传真收发的成功率，会影响语音识别的成功率。特别是语音的声纹识别成功率会降低。

第二章 VOIP NAT 穿透部分

VOIP 系统上实网上的公网应用 NAT 穿透是永远绕不过去的问题。

26. 什么是 NAT？

NAT(Network Address Translators)，网络地址转换：网络地址转换是在 IP 地址日益缺乏的情况下产生的，它的主要目的就是为了能够地址重用。NAT 分为两大类，基本 的 NAT 和 NAPT(Network Address/Port Translator)。

NAT 是一种广泛应用的解决 IP 短缺的

有效方法， NAT 将内网地址转和端口号换成合法的公网地址和端口号，建立一个会话，与公网主机进行通信。

最开始 NAT 是运行在路由器上的一个功能模块。

NAT 的产生基于如下事实：一个私有网络的节点中只有很少的节点需要与外网连接。那么这个子网中其实只有少数的节点需要外网的 IP 地址，其它的节点的 IP 地址应该是可以重用的。

因此，基本的 NAT 实现的功能很简单，在子网内使用一个保留的 IP 子网段，这些 IP 对外是不可见的。子网内只有少数一些 IP 地址可以对应到真正全球唯一的 IP 地址。如果这些节点需要访问外部网络，那么基本 NAT 就负责将这个节点的子网内 IP 转化为一个全球唯一的 IP 然后发送出去。(基本的 NAT 会改变 IP 包中的原 IP 地址，但是不会改变 IP 包中的端口)

上面描述很拗口，归纳一下可以这样理解：一个公司的局域网内的很多机器通过路由器的一个公网 IP，来使用外网上的其它服务。路由器上要做 ip 转换啥的。这样外网上的机器看到的都是这个公司我公网 ip 了。

NAT 实现地址转换，同时还起到防火墙的作用，隐藏内部网络的拓扑结构，保护内部主机。 NAT 不仅完美地解决了 IP 地址不足的问题，而且还能够有效地避免来自网络外部的攻击，隐藏并保护网络内部的计算机。 这样对于外部主机来说，内部主机是不可见的。

关于基本的 NAT 可以参看 RFC 1631

27. NAT 分类？

NAT 从表面上看有三种类型：静态 NAT 、动态地址 NAT 、地址端口转换 NAPT 。

（1 ）静态 NAT ：静态地址转换将内部私网地址与合法公网地址进行一对一的转换，且每个内部地址的转换都是确定的。

（2 ）动态 NAT ：动态地址转换也是将内部本地地址与内部合法地址一对一的转换，但是动态地址转换是从合法地址池中动态选择一个未使用的地址来对内部私有地址进行转换。

（3 ）NAPT ：它也是一种动态转换，而且多个内部地址被转换成同一个合法公网地址，使用不

同的端口号来区分不同的主机，不同的进程。

从实现的技术角度，又可以将 NAT 分成如下几类：全锥 NAT(Full Cone NAT) 、限制性锥 NAT（Restricted Cone NAT ）、端口限制性锥 NAT(Port Restricted Cone NAT) 、对称 NAT (Symmetric NAT) 。

（1 ）全锥 NAT ：全锥 NAT 把所有来自相同内部 IP 地址和端口的请求映射到相同的外部 IP 地址和端口。任何一个外部主机均可通过该映射发送数据包到该内部主机。

（2 ）限制性锥 NAT ：限制性锥 NAT 把所有来自相同内部 IP 地址和端口的请求映射到相同的外部 IP 地址和端口。但是，和全锥 NAT 不同的是：只有当内部主机先给外部主机发送数据包，该外部主机才能向该内部主机发送数据包。

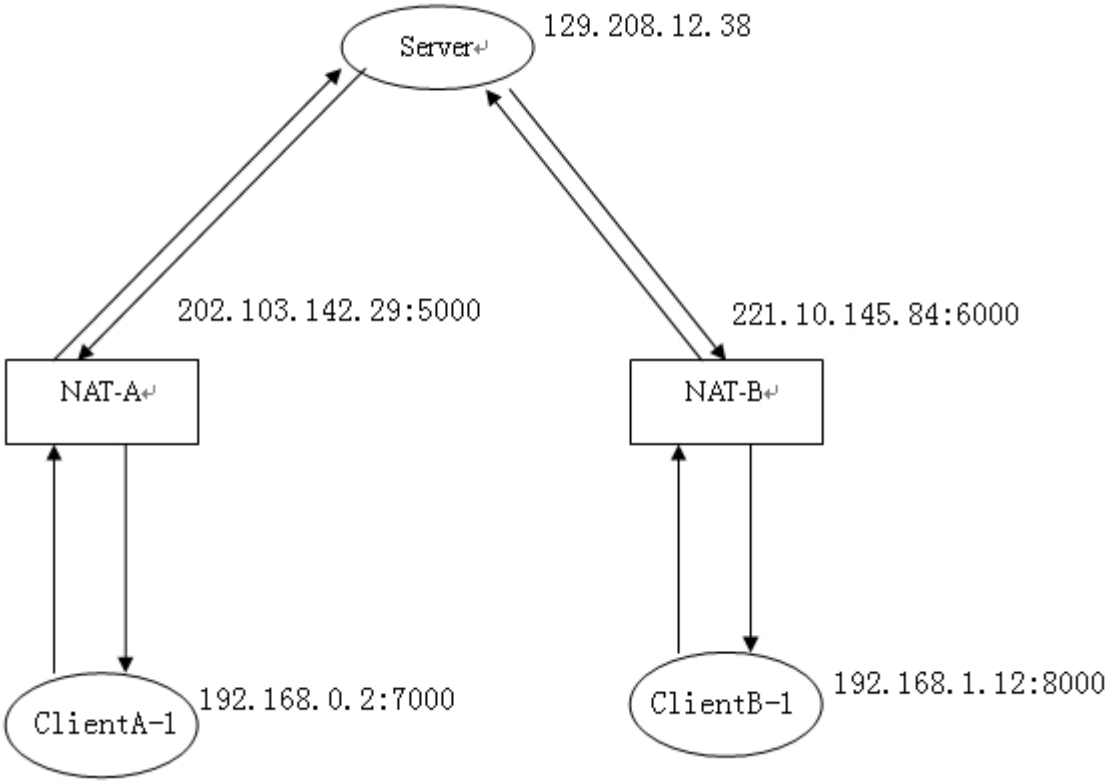
（3 ）端口限制性锥 NAT ：端口限制性锥 NAT 与限制性锥 NAT 类似，只是多了端口号的限制，即只有内部主机先向外部地址：端口号对发送数据包，该外部主机才能使用特定的端口号向内部主机发送数据包。

（4 ）对称 NAT ：对称 NAT 与上述 3 种类型都不同，不管是全锥 NAT ，限制性锥 NAT 还是端口限制性锥 NAT ，它们都属于锥 NAT （Cone NAT ）。当同一内部主机使用相同的端口与不同地址的外部主机进行通信时，对称 NAT 会重新建立一个 Session ，为这个 Session 分配不同的端口号，或许还会改变 IP 地址。

1-3 统称锥型，4 称为对称型。根据上面的描述对称型的 NAT 穿透是最麻烦的。

28. UDP 协议 NAT 穿透过程是啥？

NAT 穿透最主要目的是解决 p2p 通信的，包括 udp 通信和 tcp 通信。
要让处于 NAT 设备之后的拥有私有 IP 地址的主机之间建立 P2P 连接，
就必须想办法穿透 NAT ，现在常用的传输层协议主要有 TCP 和 UDP
VOIP 里面的 SIP、RTP 通信都是 UDP 的，我们主要关心 UDP，下面看 UDP。
根据：
http://blog.csdn.net/leisure512/article/details/4900191



参考上图，

锥（Cone ）NAT 穿透过程：

1 ．ClientA-1 （202.103.142.29:5000 ）发送数据包给 Server ，请求和 ClientB-1 （221.10.145.84:6000 ）通信。

2. Server 将 ClientA-1 的地址和端口（202.103.142.29:5000 ）发送给 ClientB-1 ，告诉 ClientB-1 ，ClientA-1 想和它通信。

3. ClientB-1 向 ClientA-1 （202.103.142.29:5000 ）发送 UDP 数据包，当然这个包在到达 NAT-A 的时候，还是会被丢弃，这并不是关键的，因为发送这个 UDP 包只是为了让 NAT-B 记住这次通信的目的地址：端口号，当下次以这个地址和端口为源的数据到达的时候就不会被 NAT-B 丢弃，这样就在 NAT-B 上打了一个从 ClientB-1 到 ClientA-1 的孔。

4. 为了让 ClientA-1 知道什么时候才可以向 ClientB-1 发送数据，所以 ClientB-1 在向 ClientA-1（202.103.142.29:5000 ）打孔之后还要向 Server 发送一个消息，告诉 Server 它已经准备好了。

5. Server 发送一个消息给 ClientA-1 ，内容为：ClientB-1 已经准备好了，你可以向 ClientB-1 发送消息了。

6. ClientA-1 向 ClientB-1 发送 UDP 数据包。这个数据包不会被 NAT-B 丢弃，以后 ClientB-1 向 ClientA-1 发送的数据包也不会被 ClientA-1 丢弃，因为 NAT-A 已经知道是 ClientA-1 首先发起的通信。至此，ClientA-1 和 ClientB-1 就可以进行通信了。

对称 NAT 和锥 NAT 很不一样。对于 对称 NAT ，当一个私网内主机和外部多个不同主机通信时，对称 NAT 并不会像锥 NAT 那样分配同一个端口。

而是会新建立一个 Session ，重新分配一个端口。参考上面穿透限制性锥 NAT 的过程，在步骤 3 时：ClientB-1 （221.10.145.84: ？）向 ClientA-1 打孔的时候，对称 NAT 将给 ClientB-1 重新分配一个端口号，而这个端口号对于 Server 、ClientB-1 、ClientA-1 来说都是未知的。

同样， ClientA-1 根本不会收到这个消息，同时在步骤 4 ，ClientB-1 发送给 Server 的通知消息中，ClientB-1 的 socket 依旧是（221.10.145.84:6000 ）。而且，在步骤 6 时：ClientA-1 向它所知道但错误的 ClientB-1 发送数据包时，NAT-1 也会重新给 ClientA-1 分配端口号。所以，穿透对称 NAT 的机会很小。

先介绍一下 STUN 协议。STUN （Simple Traversal of UDP Through NATs ）协议是一个轻量级协议，用来探测被 NAT 映射后的地址：端口。STUN 采用 C/S 结构，需要探测自己被 NAT 转换后的地址：端口的 Client 向 Server 发送请求，Server 返回 Client 转换后的地址：端口。

当 ClientB-1 收到 Server 发送给它的消息后，ClientB-1 即打开 3 个 socket 。socket-0 向 STUN Server 发送请求，收到回复后，假设得知它被转换后的地址：端口（ 221.10.145.84:600 5 ），socket-1 向 ClientA-1 发

送一个 UDP 包，socket-2 再次向另一个 STUN Server 发送请求，假设得到它被转换后的地址：端口（ 221.10.145.84:6020 ）。

通常，对称 NAT 分配端口有两种策略，一种是按顺序增加，一种是随机分配。如果这里对称 NAT 使用顺序增加策略，那么，ClientB-1 将两次收到的地址：端口发送给 Server 后，Server 就可以通知 ClientA-1 在这个端口范围内猜测

刚才 ClientB-1 发送给它的 socket-1 中被 NAT 映射后的地址：端口，ClientA-1 很有可能在孔有效期内成功猜测到端口号，从而和 ClientB-1 成功通信。

问题总结: 从上面两种穿透对称 NAT 的方法来看，都建立在了严格的假设条件下。但是现实中多数的 NAT 都是锥 NAT ， 因为资源毕竟很重要，对称 NAT ，由于太不节约端口号所以相对来说成本较高。所以，不管是穿透锥 NAT ， 还是对称 NAT ，现实中都是可以办到的。除非对称 NAT 真的使用随机算法来分配可用的端口。这个时候就无法穿透， 但是概率很小。

这个过程就是 p2p 的 udp 协议的穿透过程。

在 voip 通信中，假如媒体不需要经过服务器，而是终端直接到终端的模式（bypass 模式）,就要实现类似的穿透。

29. VOIP 服务器在公网为啥也要实现 NAT 穿透？

在 VOIP 通信上 NAT 的问题通常是因为下面 3 个原因导致：

1. VOIP 是基于 UDP 通信的，没有固定的连接，对比来看 http web，ftp 啥的是基于 tcp，有固定的连接。客户端发起通信之后通路就通了，大家可以收发数据。因此 http，ftp 啥的通常不需要 NAT 穿透。
2. VOIP 比如 sip，信令控制和媒体流是分开的。
3. VOIP 的媒体流的 ip 和端口是在 sdp 里面描述的。Sdp 是客户端在发起呼叫的时候

造好的。此时根本不知道外网的 ip 和外网将要使用端口（语音 rtp 包都还没有发呢）。因此服务端收到的 sdp 里面的 ip 和 port 都是客户端内网的 ip 和 port。将来服务器发包的时候会发到这个 ip 和 port 上，这样显然是错误的，因此需要一些办法来纠正这个问题，这个办法就是 NAT 穿透。

30. VOIP 如何实现 NAT 穿透？

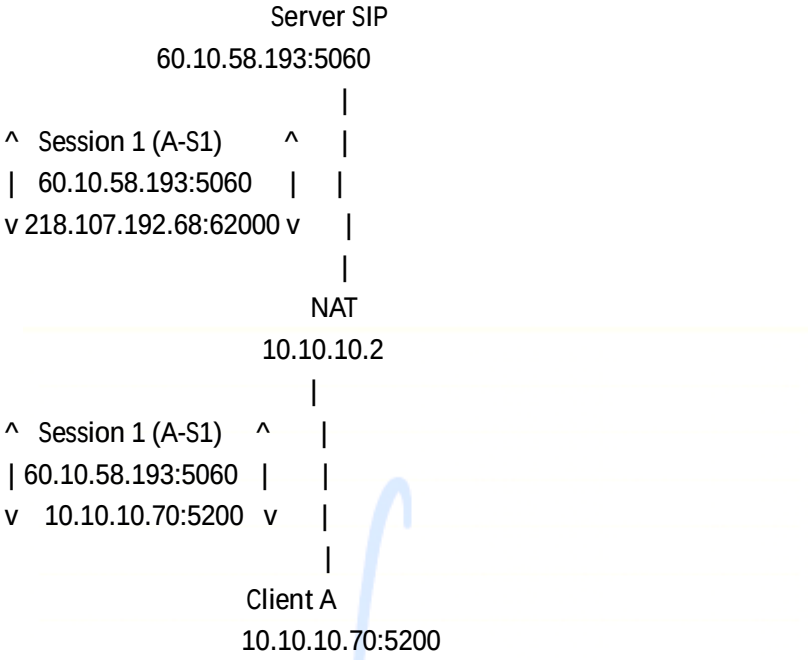
这个问题解释起来比较复杂。分为两种，第一种是媒体通过服务器（比如 FreeSwitch）的方式，第二种是媒体没有通过服务器，而是直接 p2p 的方式，这个模式参考上上个问题，需要 ice 和 stun 服务器的支持。
由于全业务的应用，服务器上可能对用户的通话进行全程录音啥的，大多数情况下，都需要媒体通过服务器。
因此这里讨论第一种方式如何实现 NAT 穿透：
这种方式实现 NAT 穿透在 VOIP 上可以这样描述：假设 VOIP 服务端，比如 FreeSwitch，在公网上，能跟在其它公司内部局域网上的 VOIP 客户端进行 VOIP 通话。
我们的希望是通过服务器，在服务器上自己就实现。
而不是借助第 3 方服务器之类的办法。

下面就就是介绍这个原理：

举例如下:

1. SIP Server IP 60.10.58.193 sip 端口是 5060
2. SIP Client :Sip 软电话 内网地址:10.10.10.70, sip 端口是 5200
3. 路由器 公司内网地址:10.10.10.2,路由器外网地址: 218.107.192.68
4. sip 通信过程:
   SIP Client 发起 sip 信令.
   SIP 里面包括 sdp 的数据协议.
   SDP 协议里面的 RTP 地址填: 10.10.10.70
   RTP PORT= 10140

SIP Server 的信令过程

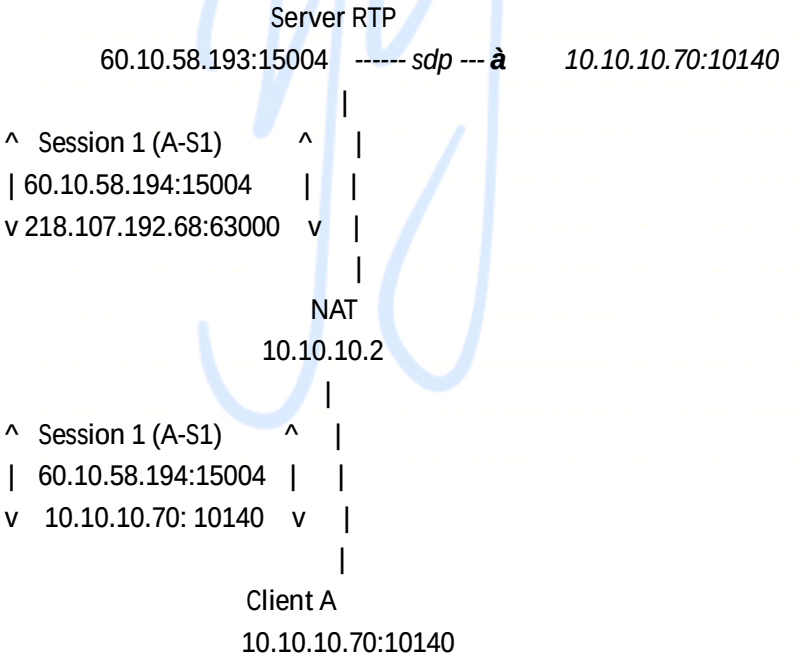


说明：经过路由器之后：sip client 的端口 5200 变成了，62000（这个是路由器分配的）

5. RTP 的话音通信过程

注意: Server RTP 原来是根据 SDP 协议里面的地址 10.10.10.70: 10140 发送 RTP,需要改为根据收到的 RTP 的来源发送 RTP.才能实现 NAT 穿透.

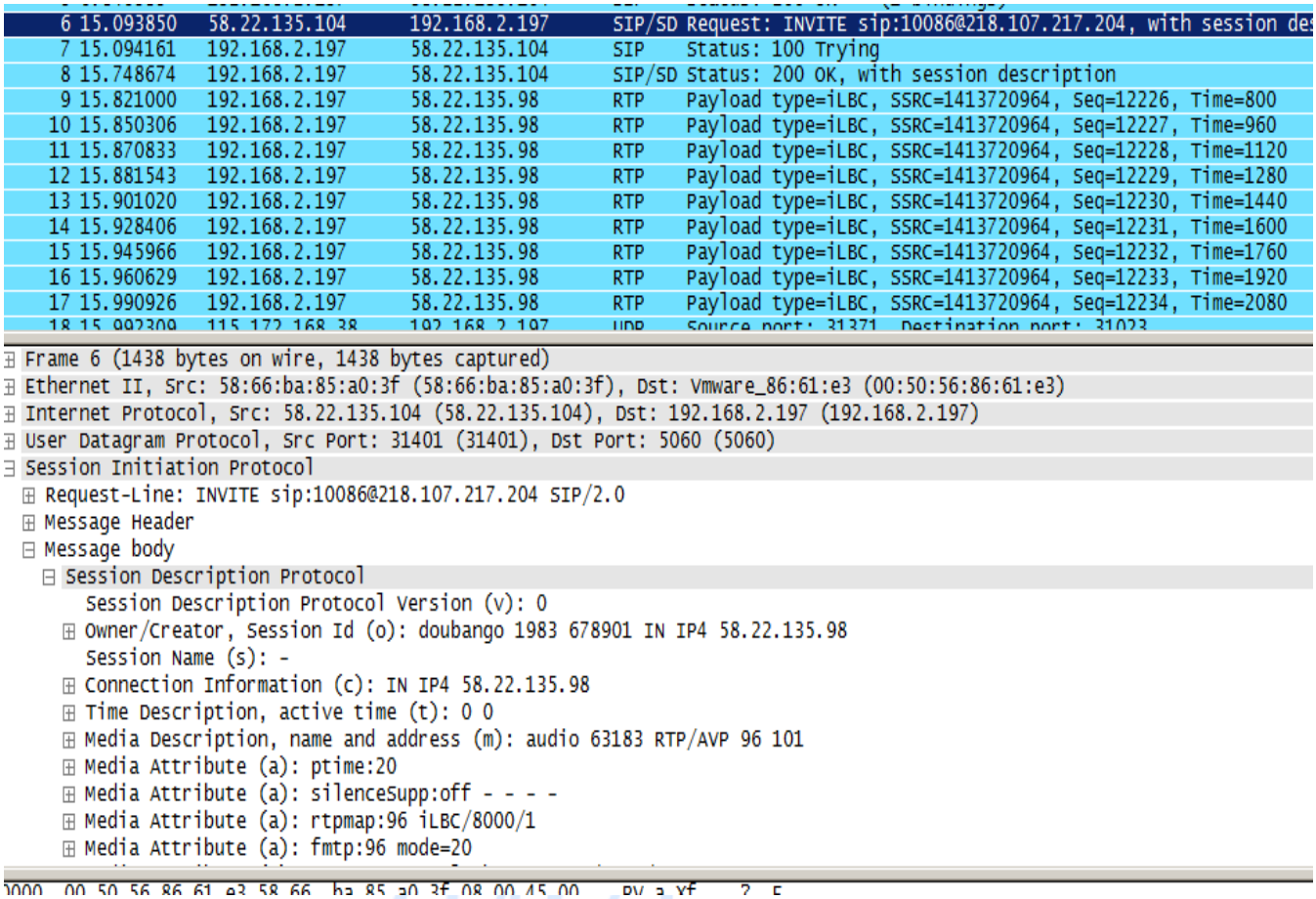
这是通过 server 上当判断收到的 rtp 的包和 SDP 里面的参数不一样时,会上报事件給应用程序,然后允许应用程序重新设置 RTP 参数才能实现的.



31. FreeSwitch 如何实现 VOIP 的 NAT 穿透?

参见下面的 eyebeam 和公网的 FreeSwitch 通信的 FreeSwitch 服务端 VOIP 抓包：

FreeSwitch 的 ip 是 192.168.2.197,VOIP 客户端的 SIP 地址是 58.22.135.104，



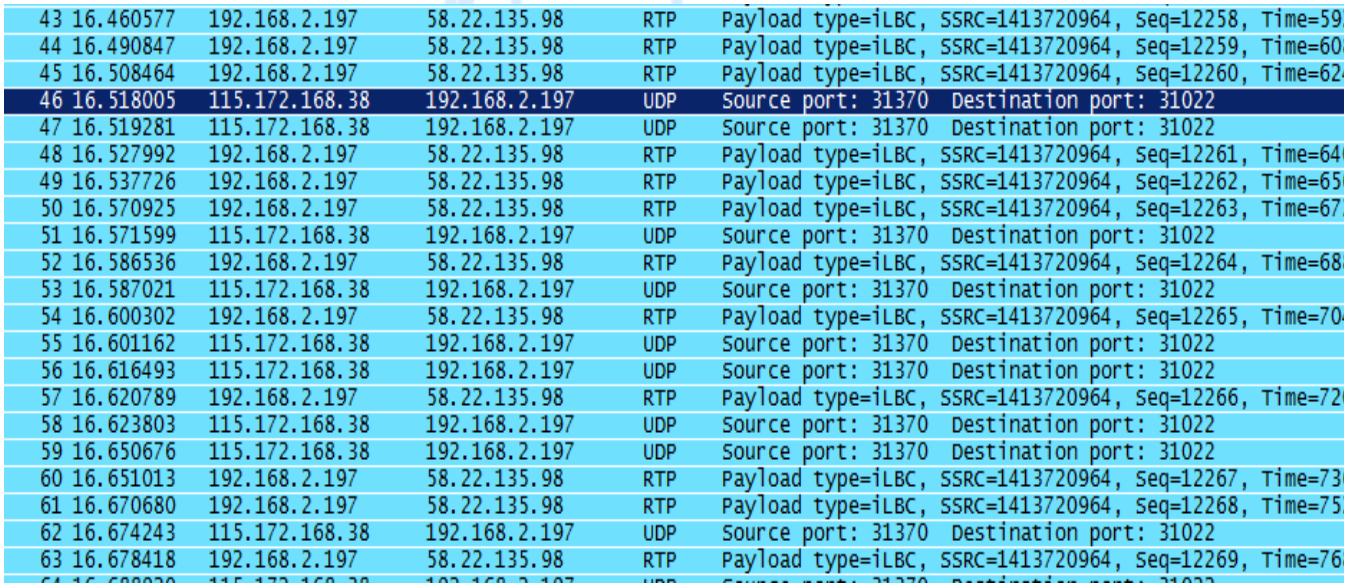
6 15.093850	58.22.135.104	192.168.2.197	SIP/SD	Request: INVITE sip:10086@218.107.217.204, with session des
7 15.094161	192.168.2.197	58.22.135.104	SIP	Status: 100 Trying
8 15.748674	192.168.2.197	58.22.135.104	SIP/SD	Status: 200 OK, with session description
9 15.821000	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12226, Time=800
10 15.850306	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12227, Time=960
11 15.870833	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12228, Time=1120
12 15.881543	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12229, Time=1280
13 15.901020	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12230, Time=1440
14 15.928406	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12231, Time=1600
15 15.945966	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12232, Time=1760
16 15.960629	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12233, Time=1920
17 15.990926	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12234, Time=2080
18 15.992309	115.172.168.38	192.168.2.197	UDP	Source port: 31371 Destination port: 31023

```
⊞ Frame 6 (1438 bytes on wire, 1438 bytes captured)
⊞ Ethernet II, Src: 58:66:ba:85:a0:3f (58:66:ba:85:a0:3f), Dst: Vmware_86:61:e3 (00:50:56:86:61:e3)
⊞ Internet Protocol, Src: 58.22.135.104 (58.22.135.104), Dst: 192.168.2.197 (192.168.2.197)
⊞ User Datagram Protocol, Src Port: 31401 (31401), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol
  ⊞ Request-Line: INVITE sip:10086@218.107.217.204 SIP/2.0
  ⊞ Message Header
  ⊟ Message body
    ⊟ Session Description Protocol
        Session Description Protocol Version (v): 0
      ⊞ Owner/Creator, Session Id (o): doubango 1983 678901 IN IP4 58.22.135.98
        Session Name (s): -
      ⊞ Connection Information (c): IN IP4 58.22.135.98
      ⊞ Time Description, active time (t): 0 0
      ⊞ Media Description, name and address (m): audio 63183 RTP/AVP 96 101
      ⊞ Media Attribute (a): ptime:20
      ⊞ Media Attribute (a): silenceSupp:off - - - -
      ⊞ Media Attribute (a): rtpmap:96 iLBC/8000/1
      ⊞ Media Attribute (a): fmtp:96 mode=20
```

VOIP SIP 发给 FreeSwitch 的 INVITE 的 SDP 的里面的媒体的 IP 是 58.22.135.98，端口是 63183

因此 FreeSwitch 发 RTP 到 58.22.135.98 了。这个时候，VOIP 客户端是听不见声音的。

后续，FreeSwitch 开始收到客户端的 rtp 的语音数据包如下：



43 16.460577	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12258, Time=59
44 16.490847	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12259, Time=60
45 16.508464	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12260, Time=62
46 16.518005	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
47 16.519281	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
48 16.527992	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12261, Time=64
49 16.537726	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12262, Time=65
50 16.570925	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12263, Time=67
51 16.571599	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
52 16.586536	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12264, Time=68
53 16.587021	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
54 16.600302	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12265, Time=70
55 16.601162	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
56 16.616493	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
57 16.620789	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12266, Time=72
58 16.623803	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
59 16.650676	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
60 16.651013	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12267, Time=73
61 16.670680	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12268, Time=75
62 16.674243	115.172.168.38	192.168.2.197	UDP	Source port: 31370 Destination port: 31022
63 16.678418	192.168.2.197	58.22.135.98	RTP	Payload type=iLBC, SSRC=1413720964, Seq=12269, Time=76

这个会话的 rtp 的语音数据包 ip 变成了 115.172168.38，端口是 31370，因此 FreeSwitch 发 RTP 到 115.172168.38 了。然后 115.172168.38 这个 ip 上的路由器把包转发给内网的 voip 客户端，这个时候，VOIP 客户端就可以听见声音了。一句话归纳是：服务端的语音包要发到哪里去是要能跟随哪里来语音包进行变动，才能支持 NAT 穿透。或者简单一点:哪里去 by 哪里来。

最难的 NAT 应用是在 NAT 环境下的 P2P（媒体没有经过服务器）的通信。本章没有解决这个问题。

第三章 FreeSwitch 基础和配置部分

32. FreeSwitch 是什么？

有人说 FreeSwitch =Free+Switch=自由免费+交换通信
他是一个开源的电话交换平台，它具有很强的可伸缩性，从一个简单的软电话客户端到运营商级的软交换设备。它可以是一个 SIP SERVER，也可以通过它实现很多协议转换。也可以实现 VOIP 的 IVR 或者呼叫中心。我就使用他作为基础开发了一个 CTI 平台和一个呼叫中心应用平台。

33. FreeSwitch 是谁发起开发的？

一个真正的牛人叫做 Anthony Minessale 的伙计在 2005 年的时候认为 Asterisk 存在许多问题而修复这些问题需要很多时间。于是他想从头创建一个 Asterisk 2.0 后面就变成了 FreeSwitch，因此从某种意义上说 FreeSwitch 是 Asterisk 2.0。一开始没有人认真考虑他的问题，因此他就自己开发，真正牛人通常就是这样。呵呵
从中我们知道两个：
一个是以前 Asterisk 有不少问题，不够健壮，所以搞 Asterisk 的同学们这些惨了，站错了队了，赶紧跳槽到 FreeSwitch 来吧。还好据说新版本的 AS 已经 比较稳定。:-）
另外一个是 FreeSwitch 跟 Asterisk 在很多地方很像，也可以说是山寨的 Asterisk。

34. FreeSwitch 历史是什么？

2005 开始怀孕
2007 年发表 1.0
2010 年发布 1.0.6
2012 年发布 1.2.X 和 1.3.X
2013 年发布 1.2.XX
2014 年发布 1.2.23 和 1.4.7
跟其它的开源的 SIP 比较起来，其实 FreeSwitch 比较年轻。
年轻就是说长的快因此版本更新比较快。上面说过搞开源的一个原则是千万不能选那种没有更新的开源软件。

35. FreeSwitch 能做啥？

FreeSwitch 几乎无所不能可以用作，一个简单的交换引擎、一个 PBX，
一个媒体网关或媒体支持 IVR 的服务器等。
它支持 SIP、H323、Skype、Google Talk，RTMP 等协议，支持板卡 E1 接口，这样就可以实现通过运营商打电话到手机或者固定电话之上。

36. FreeSwitch 如何与其它系统集成？

FreeSwitch 跟其它系统集成基本上是通过它的支持的协议接口进行集成的
最主要的集成方式就是通过 SIP 协议。
可以作为一个分机注册到其它系统上，也可以其它系统作为分机注册到 FreeSwitch 上以实现互通，甚至可以不通过注册直接使用点对点方式进行通讯。
本人实际使用过程中：集成过以下第 3 方的系统或者设备
FreeSwitch==SIP==毅航公司 ISX1000/4000/6000 系列多媒体可编程交换机
ISX1000/4000/6000 系列的新驱动支持 iLBC 编码器，可以和 FreeSwitch 公网集成，
从而实现 FreeSwitch 的落地出口。
FreeSwitch==SIP==东进公司 Keygoe 系列多媒体可编程交换机
FreeSwitch==SIP==Dialogic 公司 HMP 多媒体可编程软交换系列
FreeSwitch== SIP==迅时模拟网关===模拟电话线==PSTN
FreeSwitch== SIP==鼎信通达数字 E1 网关===E1==PSTN
FreeSwitch== SIP==网经数字 E1 网关===E1==PSTN

37. FreeSwitch 最新版稳定本号是什么？

在 http://files.FreeSwitch.org/上最新的稳定版本是 1.2.23
本人测试这个版本单机 2000 线并发基本上可以使用。
还有一个版本是 1.4.7，这个是支持 WebRTC 的版本，
在 1.4.X 的正式版没有发布之前，官方建议使用 1.2.X 的稳定版本，
在 1.4.X 的正式版本出来之后，官方建议使用投产系统 1.4.X 的版本。

38. FreeSwitch 支持哪些操作系统？

主流的 LINUX 和 WINDOWS 操作系统都支持。都有现成的安装包。
这个对初学者比较有利。
很多初学者不熟悉 LINUX 系统，使用 WINDOWS 可以迅速入门，安装配置到运行起来
呼叫通第一个软化电话前后不超过 10 分钟。会比较有成就感;-) !
而且可以是绿色版本，自己编译的版本，拷贝一下就可以使用，无需安装配置。

39. 去哪里下载 FreeSwitch 安装包和源码？

在 http://files.FreeSwitch.org/ 下载打包好的 tar.gz 文件比较方便，初学者可以现在已经编译好的二进制安装包。
使用 GIT 去下载源码很麻烦。另外最新的版本经常有莫名其妙的问题。
因此不建议。

Windows 版本有编译好的二进制文件，
32 位版本
http://files.freeswitch.org/windows/installer/x86/freeswitch.msi
64 位版本
http://files.freeswitch.org/windows/installer/x64/freeswitch.msi
凑合使用是可以，好处是不需要编译对初学者比较合适。
正式使用，或者老鸟最好自己下载源码编译。

40. FreeSwitch 在 windows 下如何安装？

以 1.2.3 版本为例，下载 FreeSwitch.msi
然后双击运行，然后点 next
然后在 END-User License 界面 勾选 I Accept。。。
然后点 next，然后选择 Complete 完全安装（反正没有多大），然后点 Install
就开始安装拷贝。 拷贝完成 点 ok 完成安装。
假如机器上有 360 之类的还要点允许访问和记住选择。
FreeSwitch 会安装到 C:\Program Files\FreeSwitch 目录下
假如机器上之前安装过去其它版本，请先卸载，然后完全删
除 C:\Program Files\FreeSwitch 目录 以及目录下的所有东西，然后再安装，
假如 C:\Program Files\FreeSwitch 目录 以及目录下的有文件东西将可能导致
安装不成功

41. FreeSwitch 在 LINUX 下如何编译和安装？

以下是本人在 centos 6.3 上的安装步骤：
install FreeSwitch on centos 6.3:

1）先到 http://files.FreeSwitch.org/ 下载 源码包 FreeSwitch-1.2.5.3.tar.bz2
2）然后 tar jxvf FreeSwitch-1.2.5.3.tar.bz2
假如是.gz 结尾的请使用-z 命令： tar zxvf FreeSwitch-1.2.4.tar.gz

3）核查必须安装工具：
sudo yum install git autoconf automake libtool ncurses-devel libjpeg-devel
4）核查可选的安装工具：
sudo yum install expat-devel openssl-devel libtiff-devel libX11-devel unixODBC-devel libssl-devel python-devel zlib-devel libzrtpcpp-devel alsa-lib-devel libogg-devel libvorbis-devel perl-libs gdbm-devel libdb-devel uuid-devel @development-tools

5) ./rebootstrap.sh
6) ./configure
7) vi modules.conf 添加需要的额外模块比如 mod_rtmp 模块 默认 是没有 not found rtmp endpoint 和 ilbc codec 两个模块的
8) make
9) make all install cd-sounds-install cd-moh-install

10) cd /usr/local/freeswitch/conf/autoload_configs

vi modules.conf.xml 根据需要增加相应的模块，比如 mod_rtmp 和 ilbc 模块

11) cd /usr/local/freeswitch/bin

./freeswitch

说明：

默认 FreeSwitch 安装在/usr/local/FreeSwitch/，然而如果不想使用默认安装路径，可以在 ./configure 的时候 添加“--prefix=”参数改变 FreeSwitch 的安装路径。

./configure --prefix=/usr/FreeSwitch

42. FreeSwitch 在 windows 下如何编译？

1.2.1 或者之后的版本需要使用 VS2010 进行编译。

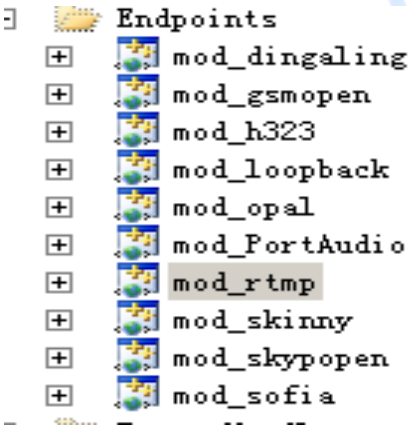
本人使用 VS2010 旗舰版（安装 VS2010 的时候 C#也要安装，否则有些 FreeSwitch 的工程无法打开）。

VS2010 运行之后，打开源码包里面的 FreeSwitch.2010.sln。打开之后可能会出现

点确定不管它。然后开始编译。

注意：早期的版本，有许多模块，比如 rtmp_mod 模块默认是没有编译产生的，需要手工自己在模块对应的工程右键点编译。

如下图：



或者在配置管理器里面勾选对应的模块

如下图：

编译之后会在工程文件所在的目录下产生

\Win32\Debug 目录 （Debug 版本）

或者 Win32\Release 目录（Release 版本）

它们的目录下包括了运行的 exe 文件 dll 和对应的配置文件目录（跟安装包安装产生的文件基本一样，额外多了一下 pdb 之类的文件）。

直接运行 Debug 或者 Release 目录下的 FreeSwitchConsole.exe

就可以启动

1.2.14 之后的版本基本上默认编译就带该有的都有了，不需要手工选择。

43. FreeSwitch 在 windows 下如何安装到 C 盘之外？

Windows 版本的安装包只能安装到 C:\Program Files\FreeSwitch 这个目录下，而且变态的安装包不能修改安装目录。

要搞到其它目录运行可以在安装之后，拷贝整个 FreeSwitch 目录到其它盘比如 d:

然后运行 d:\ FreeSwitch 目录下的 FreeSwitchConsole.exe。

这个说明 FreeSwitch 是没有环境变量绿色版本。

作为程序员的偶很喜欢这个的绿色版。不需要安装，不需要重启，部署的时候直接整个目录一锅端就 ok 了。

44. FreeSwitch 在实际使用部署的时候如何启动比较安全？

带上-nonat 参数启动比较安全,另外一个福利是比较快。

FreeSwitchConsole.exe -nonat

带上这个参数之后启动不需要到路由器上去检查外网地址啥的，因此启动比不带-nonat 的启动方式快了很多，启动通常在 10 秒内完成。默认的启动屏幕会输出很多日志，而且有各种颜色，假如有看到红色的日志要认真看，因为错误才会输出红色。这个时候需要查是啥错误导致。免得将来碰到莫名其妙的问题。

45. FS_CLI 跟 FreeSwitch 是啥关系？

FS_Cli 是 FreeSwitch 一个客户端控制界面，是通过 esl lib 连接到 fs 的 esl 模块上进行控制的。
将来自己开发程序也是这个模式，这样的意思是自己开发完全可以参考 fs_cli 的代码。尤其是当碰到问题不行的时候，先使用 fs_cli 测试，假如 fs_cli 测试可以，而自己开发的软件不行，那就看看 fs_cli 里面是咋实现的。本人就是通过这个方法来解决 esl 开发过程中的问题。
可以在 FS_Cli 上对 FreeSwitch 进行管理，比如日志级别设置，查看日志，执行呼叫等操作。FS_Cli 是通过 ESL 接口对 FreeSwitch 进行管理。FS_Cli 也可以执行 APP 模拟进行发起呼叫，播放语音等功能。
FS_Cli 有快捷按键，F1-F12,功能对应如下:

```
<key name="1" value="help"/>
<key name="2" value="status"/>
<key name="3" value="show channels"/>
<key name="4" value="show calls"/>
<key name="5" value="sofia status"/>
<key name="6" value="reloadxml"/>
<key name="7" value="console loglevel 0"/>
<key name="8" value="console loglevel 7"/>
<key name="9" value="sofia status profile internal"/>
<key name="10" value="sofia profile internal siptrace on"/>
<key name="11" value="sofia profile internal siptrace off"/>
<key name="12" value="version"/>
```

FS_Cli 默认是连接到本机的机器上，也可以连接到其它机器上的 FreeSwitch 进行管理。

46. 在 FS_CLI 上如何拨打测试分机？

在 FreeSwitch 控制台或者 FS_CLI 的界面上可以下控制命令进行外拨：

比如：拨打 1000 并执行 echo 程序

```
FreeSwitch> originate user/1000 &echo
```

拨打 1000 分机，然后转接到 1001，相当于在软电话 1000 上拨打 1001

```
FreeSwitch> originate user/1000  1001 XML default
```

47. FreeSwitch 能跟哪些外部协议对接？

它支持 SIP、H323、Skype、Google Talk，RTMP ，WebRTC 等协议
实际本人测试过 sip ，RTMP 和 WebRTC。
其它协议未测试。

48. FreeSwitch 如何跟 PSTN 对接，实现落地？

有两个办法：
办法 1. FreeSwitch 通过 SIP 接到第三方的 VOIP 网关上，VOIP 网关通过 E1 接口接到 PSTN 上，通常 VOIP 网关可以支持 ISDN PRI 和 SS7 信令。比如上面提到的通过 ISX1000/4000,或者迅时网关 或者鼎信通达的网关进行落地。

办法 2.使用支持 FreeSwitch 的 E1 接口卡，在机器上插这种卡，安装卡驱动。然后安装 FreeSwitch ，再进行协议配置才能使用。假如使用 ISDN 协议，推荐使用 Sangoma 板卡。有 pci 和 pci-e 两个接口。

49. 已经有哪些硬件板卡支持 FreeSwitch 跟运营商的 E1 电路对接？

FreeSwitch 的 FreeTDM 模块是 sangoma 公司开发维护的，
因此使用 sangoma 板卡是最合适的。Sangoma 卡可以 ISDN PRI 和 SS7 信令。

如何在 LINUX 的 FreeSwitch 平台上安装 sangoma 请参考：
http://www.voip88.com/article-1202-1.html
如何安装在 Windows 的 FreeSwitch 平台上安装 sangoma 请参考：
http://wiki.sangoma.com/FreeSwitch-windows-FreeSwitch-compile-isdn
SS7 信令 的支持软件是商业版本，是要额外收费的，根据了解，1 个 E1 端口差不多
是 1K 的授权费用，
根据 http://wiki.sangoma.com/wanpipe-FreeSwitch#sangoma-freetdm-ss7-library-libsngss7
上面的介绍，ss7 不支持 tup，仅仅支持 isup：

Sangoma FreeTDM SS7 Library (libsng_ss7)

Features

Sangoma's SS7 Library uses Continuous Computing's (Trillium) MTP2/3 and ISUP stacks to provide a commercial grade SS7 interface to FreeSwitch, via the FreeTDM channel driver.

List of supported variants

- ISUP (ITU/ANSI)
- MTP3 (ITU/ANSI)
- MTP2 (ITU/ANSI)

SCCP API

50. FreeSwitch 默认配置启动之后占用哪些端口？

FreeSwitch 默认配置启动之后，占用以下这些端口：
SIP 5060 5080
RTP:16384-32768
TCP：1935 假如启动 mod_rtmp 模块
Esl 的模块 8021 等
具体如下：

```
TCP     0.0.0.0:1935
TCP     127.0.0.1:8021
TCP     192.168.11.105:5060
TCP     192.168.11.105:5080
UDP     192.168.11.105:5060
UDP     192.168.11.105:5080
```

假如这些端口已经被占用，将可能导致启动错误。
其中 RTP:16384-32768 这个段太夸张，通常需要修改，
修改是通过 conf\autoload_configs\switch.conf.xml 里面的下面两个参数：

```
<param name="rtp-start-port" value="16384"/>
<param name="rtp-end-port" value="32768"/>
```

51. 如何看 FreeSwitch 启动之后 sip 模块是否工作正常？

FreeSwitch 启动之后， 会使用某个 ip 的 5060 端口进行侦听，
假如不是在你预期的 ip 上侦听，你的软电话就注册不上去了。
如何看是否在某个 ip 的某个端口上工作正常,可以通过 sofia status 命令看：

```
FreeSwitch@yhy-PC32> sofia status

Name                Type                                    Data          State
=================================================================================
192.168.1.102               alias                                   internal        ALIASED
    internal        profile             sip:mod_sofia@192.168.1.102:5060        RUNNING (0)
    external        profile             sip:mod_sofia@192.168.1.102:7080        RUNNING (0)
   external::example.com    gateway                 sip:joeuser@example.com         NOREG
  external::gw2         gateway                 sip:5678@192.168.1.107:7060         NOREG
  external::gw1         gateway                 sip:5678@192.168.2.223:5060         NOREG
=================================================================================
2 profiles 1 alias
```

上面的结果说明：

internal 的 sip 模块在 192.168.1.102:5060 工作，
external 的 sip 模块在 192.168.1.102:7080 上工作。
当然还有网关啥的。可以暂时不管这个。

52. FreeSwitch 为啥会一直运行在 127.0.0.1 的 IP 上？

本人测试，假如只有一个网卡，网卡上只有配置了一个 IP，但是网卡上没有配置网关，或者网卡上没有接线。
FreeSwitch 是无法使用这个网卡的 ip，于是就使用 127.0.0.1 的 ip 作为 sip 服务 5060 的端口地址，从而导致其它机器无法注册上来。
碰到这种情况，sofia status 命令 查看，或者
可以使用 netstat –ona 先看看 udp 5060 port 的进程是哪个占用。占用的 Ip 地址是哪个。

53. 如何指定 FreeSwitch 运行在没有配置网关的 IP 上？

上面说过：假如网卡的 ip 没有设置网关。FreeSwitch 变态就不使用这个 ip 作为服务的 ip。
而使用 127.0.0.1 作为服务 ip。
要使用这个没有网关的 ip 假设是 192.168.1.101 作为服务 ip 需要修改两到三个地方：

1）修改 conf\vars.xml 里面的

```
<X-PRE-PROCESS cmd="set" data="domain=$${local_ip_v4}"/>
```

改为 `<X-PRE-PROCESS cmd="set" data="domain=192.168.1.101"/>`

2)conf\sip_profiles\internal.xml 里面的

```
<param name="rtp-ip" value="$${local_ip_v4}"/>
<param name="sip-ip" value="$${local_ip_v4}"/>
```

改为

```
<param name="rtp-ip" value="192.168.1.101"/>
<param name="sip-ip" value="192.168.1.101"/>
```

3）假如使用到 external 还需要修改
conf\sip_profiles\external.xml 里面的

```
<param name="rtp-ip" value="$${local_ip_v4}"/>
<param name="sip-ip" value="$${local_ip_v4}"/>
```

改为

```
<param name="rtp-ip" value="192.168.1.101"/>
<param name="sip-ip" value="192.168.1.101"/>
```

54. FreeSwitch 如何修改默认 SIP 端口？

FreeSwitch 启动之后，默认的 internal 的 SIP 端口 5060,假如公网部署，这个端口会经常被人攻击，

甚至被盗打电话，假如要修改，修改 conf\下的
vars.xml 文件里面：

```
<X-PRE-PROCESS cmd="set" data="internal_sip_port=5060"/>
```

external 的默认 SIP 端口 是 5080 假如要修改，修改 vars.xml 文件里面以下参数：

```
<X-PRE-PROCESS cmd="set" data="external_sip_port=5080"/>
```

更加要命的是 5080 的拨入不需要认证，因此需要修改：

```
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
```

为

```
<X-PRE-PROCESS cmd="set" data="external_auth_calls=true"/>
```

55. FreeSwitch 如何修改默认 RTP 端口范围？

FreeSwitch 启动之后，默认的 RTP 端口 范围是 16384-32768 占用以下这些端口：
这个范围太大，有点夸张，假如要修改可以通过修改：
conf\autoload_configs\ switch.conf.xml 配置文件里面的：

```
<!-- RTP port range -->
<param name="rtp-start-port" value="10000"/>
<param name="rtp-end-port" value="20000"/>
```

修改之后重启 FreeSwitch，就可以。

56. FreeSwitch 在多个 IP 机器上如何指定运行在某个 IP 上？

修改\conf\sip_profiles\ internal.xml 和 external.xml

```
<param name="sip-ip" value="$${local_ip_v4}"/>
<param name="sip-port" value="$${internal_sip_port}"/>
<param name="rtp-ip" value="$${local_ip_v4}"/>
```

对应于 internal.xml 和 external.xml 的 sip 的 ip 和 rtp 的 ip
另外一个办法是：也可以直接在 vars.xml 里面指定

```
<X-PRE-PROCESS cmd="set" data="local_ip_v4=192.168.1.169"/>
<X-PRE-PROCESS cmd="set" data="domain_name=$${domain}"/>
```

57. FreeSwitch 常用目录有哪些？

主要目录：

mod 可加载模块
sounds 声音文件,使用 playback() 时默认的寻找路径
log 日志，CDR 等
recordings 录音，使用 record() 时默认的存放路径

conf 配置文件目录
scripts 脚本的目录，比如 lua 脚本就存放在这个目录下。

58. FreeSwitch 基本配置文件有哪些？

在/Conf 目录下：
vars.xml 文件：一些常用变量默认分机密码=1234，codec，sip ，ip，port 等
/sip_profiles
/autoload_configs
/dialplan
/directory
/dialplan/default.xml 缺省的拨号计划
/directory/default/*.xml SIP 用户，每用户一个文件
/sip_profiles/internal.xml 一个 SIP profile，或称作一个 SIP-UA，
监听在本地 IP 及端口 5060，一般供内网用户使用
/sip_profiles/externa.xml 另一个 SIP-UA，用作外部连接，端口 5080
/autoload_configs/modules.conf.xml 配置当 FreeSwitch 启动时自动装载哪些模块
这些 XML 配置文件 7788 加起来快 200 个。

59. FreeSwitch 如何设置日志级别？

FreeSwitch 的日志分为两种：
a.一种是显示在界面上的日志修改：
在 FS_CLI 管理界面上：
FreeSwitch>console loglevel 级别
级别 从 0-7， 比如 6 设置成 INFO 级别
基本越高 日志越大比如设置成 7，DEBUG 级别。几乎每个操作都很多日志。输入之后，会返回当前的级别提示如下：

```
FreeSwitch> console loglevel 0
+OK log level 0 [0]
+OK console log level set to CONSOLE

FreeSwitch> console loglevel 7
+OK log level 7 [7]
+OK console log level set to DEBUG

FreeSwitch> console loglevel 6
+OK log level 6 [6]
 +OK console log level set to INFO
```

假如要看 sip 的详细日志，使用以下命令：
打开 sip 日志:
sofia profile internal siptrace on
关闭 sip 日志:

sofia profile internal siptrace off

假如要一开始就设置日志级别，比如默认启动级别改为 4（warnning），需要 修改 vars.xml 文件里面：

<X-PRE-PROCESS cmd="set" data="console_loglevel=4"/>

b.另外一种是在\log 下输出的文件日志：比如要修改为 wanning，需要修改需要一下文件 conf\autoload_configs\logfile.conf.xml，将里面的

<map name="all" value="debug,info,notice,warning,err,crit,alert"/>

改为：

<map name="all" value="warning,err,crit,alert"/>

注意：

conf\autoload_configs\switch.conf.xml 文件中控制着所有的日志输出级别

<!-- Default Global Log Level - value is one of debug,info,notice,warning,err,crit,alert -->

<param name="loglevel" value="debug"/>

假如这个地方修改了，比如修改为 crit，其他地方改了也无效。

60. FreeSwitch 如何看有多少用户注册上来？

在 FS_CLI 管理界面上：

FreeSwitch>sofia status profile internal

(显示多少用户已注册）

假如 刚刚启动，没有人注册上来：

提示如下：

。。。。。

ZRTP-PASSTHRU	false
AGGRESSIVENAT	false
STUN-ENABLED	true
STUN-AUTO-DISABLE	false
CALLS-IN	0
FAILED-CALLS-IN	0
CALLS-OUT	0
FAILED-CALLS-OUT	0
REGISTRATIONS	0

我使用 eyebeam 测试，实际上开了两个软电话，注册两个上来。控制台看到是 2 注册上来。如下图：

```
freeswitch@internal> sofia status profile internal
================================================================================
Name                    internal
Domain Name             N/A
Auto-NAT                false
DBName                  sofia_reg_internal
Pres Hosts              192.168.11.105,192.168.11.105
Dialplan                XML
Context                 public
Challenge Realm         auto_from
RTP-IP                  192.168.11.105
SIP-IP                  192.168.11.105
URL                     sip:mod_sofia@192.168.11.105:5060
BIND-URL                sip:mod_sofia@192.168.11.105:5060
HOLD-MUSIC              local_stream://moh
OUTBOUND-PROXY          N/A
CODECS IN               iLBC,PCMA
CODECS OUT              iLBC,PCMA
TEL-EVENT               101
DTMF-MODE               rfc2833
CNG                     13
SESSION-TO              0
MAX-DIALOG              0
NOMEDIA                 false
LATE-NEG                false
PROXY-MEDIA             false
ZRTP-PASSTHRU           false
AGGRESSIVENAT           false
STUN-ENABLED            true
STUN-AUTO-DISABLE       false
CALLS-IN                0
FAILED-CALLS-IN         0
CALLS-OUT               0
FAILED-CALLS-OUT        0
REGISTRATIONS           2
```

最后的一行可以看到 数目是 2，表示 2 注册上来。
这里面还有 sip 的 ip，拨入和拨出系统的编码器等等很多有用信息。
你要是认为上面的命令太臭长。
简单一点 show registrations 回车。
或者命令您老也记不住 show reg 然后 tab 键 然后回车。Fs 带了命令补全功能。

61. FreeSwitch 如何看有哪些用户注册上来？

在 FS_CLI 管理界面上：
显示哪些用户已注册：
FreeSwitch> sofia status profile internal reg
刚刚启动 FreeSwitch
FreeSwitch> sofia status profile internal reg
+OK log level [7]
FreeSwitch@internal> sofia status profile internal reg
Registrations:
==========

Total items returned: 0

启动两个软电话之后，

FreeSwitch> sofia status profile internal reg　提示如下：

```
freeswitch@internal>  sofia status profile internal reg

Registrations:
================================================================================================
Call-ID:        b032ed6b0f691432
User:           1003@192.168.11.105
Contact:        1003 <sip:1003@192.168.11.103:5060>
Agent:          eyeBeam release 3006o stamp 17551
Status:         Registered(UDP)(unknown) EXP(2012-11-22 20:39:23) EXPSECS(3587)
Host:           yhy-PC32
IP:             192.168.11.103
Port:           5060
Auth-User:      1003
Auth-Realm:     192.168.11.105
MWI-Account:    1003@192.168.11.105

Call-ID:        065c332f463e3a7f
User:           1000@192.168.11.105
Contact:        1000 <sip:1000@192.168.11.105:6060>
Agent:          eyeBeam release 3006o stamp 17551
Status:         Registered(UDP)(unknown) EXP(2012-11-22 20:40:29) EXPSECS(3653)
Host:           yhy-PC32
IP:             192.168.11.105
Port:           6060
Auth-User:      1000
Auth-Realm:     192.168.11.105
MWI-Account:    1000@192.168.11.105

Total items returned: 2
================================================================================================
```

这个可以看到注册上来 的机器的 ip 地址分机号码等详细信息。

我使用 eyebeam 测试，实际上注册两个上来。使用控制台看到也是 2 个用户注册上来。

\db\ sofia_reg_internal.db 里面保存的是注册的信息。

假如碰到意外情况，比如 FreeSwitch 被异常关闭，可能存在有软电话注册上来之后，一直在里面的情况。请看这行：

Status:　　Registered(UDP)(unknown) EXP(2012-11-22 20:45:46) EXPSECS(3655)

最后的秒数目 就是注册有效期。

假如异常的时候 EXPSECS(3655) 可能是负的比如 EXPSECS(-355)。

碰到这个问题，暴力把 fs 关闭你，把 db 目录下的 db 文件都删除就可以了。

62. FreeSwitch 如何踢掉注册上来的用户？

比如 FreeSwitch 的 ip 是 192.168.1.101，要踢掉分机 1007 在 FS_CLI 管理界面上：

FreeSwitch> sofia profile internal flush_inbound_reg　1007@192.168.1.101

+OK flushing all registrations matching specified call_id

然后使用:sofia status profile internal reg 1007 进行验证：

FreeSwitch> sofia status profile internal reg 1007

Registrations:

==========

Total items returned: 0

==========

看输出的确被踢掉了。

63. FreeSwitch 默认配置启动之后有哪些默认注册用户和密码是多少？

FreeSwitch 默认配置启动
默认有 1000-1019 总共 20 个帐号
它们的默认密码是 1234.
使用软电话可以注册上去进行呼叫。

64. FreeSwitch 如何设置不需要密码认证？

conf\sip_profiles\ internal.xml 里面的：

```
<param name="accept-blind-reg" value="true"/>
<param name="accept-blind-auth" value="true"/>
```

修改之后使用软电话可以以任何号码和密码注册上去，但是假如没有帐号，呼叫是不行的。

65. FreeSwitch 如何设置随便帐号都能使用系统？

有某些情况下，比如开放给用户测试，这个时候可以允许用户自己随便使用自己喜欢的帐号使用系统，比如使用自己的手机号码进行注册使用系统。
假如要允许系统没有的用户可以随便注册和呼叫有一种办法是使用下面 lua 脚本的办法：
先修改 conf\autoload_configs\lua.conf.xml 文件，结果如下：

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
   <param name="xml-handler-script" value="gen_dir_user_xml.lua"/>
   <param name="xml-handler-bindings" value="directory"/>
  </settings>
</configuration>
```

然后创建 gen_dir_user_xml.lua 文件并且存到 \scripts 目录下，
gen_dir_user_xml.lua 内容如下：

```
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, provided params:\n" .. params:serialize() .. "\n")

local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
local req_user   = params:getHeader("user")
  XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
  <section name="directory">
```

```
        <domain name="]] .. req_domain .. [[">
          <user id="]] .. req_user    .. [[">
            <params>
              <param name="password" value="]] .. req_user .. [["/>
              <param name="vm-password" value="]] .. req_user .. [["/>
              <param name="dial-string" value=
"{sip_invite_domain=${dialed_domain},presence_id=${dialed_user}@${dialed_domain}}${sofia_contact(${dialed_user}@
${dialed_domain})}"/>
            </params>
             <variables>
            <variable name="toll_allow" value="domestic,international,local"/>
            <param name="accountcode" value="]] .. req_user .. [["/>
            <variable name="user_context" value="default"/>
              <param name="effective_caller_id_number" value="]] .. req_user .. [["/>
            <param name="effective_caller_id_name" value="Extension ]] .. req_user .. [["/>
            <variable name="outbound_caller_id_name" value="Extension 5678"/>
            <variable name="outbound_caller_id_number" value="5678"/>
            <variable name="callgroup" value="techsupport"/>
    </variables>
          </user>
        </domain>
      </section>
</document>]]
    -- comment the following line for production:
    freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, generated XML:\n" .. XML_STRING .. "\n")
```

之后使用随便的分机号码然后使用跟分机号码一样的密码注册 就可以使用系统。

66. FreeSwitch 默认配置启动之后有哪些分机比较有用？

FreeSwitch 默认配置启动之后

下面的这些分机会经常用到：

9196	echo，回音测试 ,只有一个软电话分机的时候使用这个测试。 比如拨入测试网络是否正常，软电话的耳麦喇叭是否正常。
9195	echo，回音测试，延迟 5 秒，其他同上。
5000	示例 IVR
30xx	电话会议比如 3000，3001.拨入之后假如是第一个人会听音乐。

67. FreeSwitch 如何手工添加分机？

比如要添加一个 13606060253 的 11 位分机。

首先在 conf/directory/default 目录下 增加一个用户配置文件，配置文件可以参考已经有的配置文件。比如 建立一个 13606060253.xml 内容如下：

```
<include>
```

```
  <user id="13606060253">
    <params>
      <param name="password" value="13606060253"/>
      <param name="vm-password" value="13606060253"/>
    </params>
    <variables>
      <variable name="toll_allow" value="domestic,international,local"/>
      <variable name="accountcode" value="13606060253"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Extension 13606060253"/>
      <variable name="effective_caller_id_number" value="13606060253"/>
      <variable name="outbound_caller_id_name" value="$${outbound_caller_name}"/>
      <variable name="outbound_caller_id_number" value="$${outbound_caller_id}"/>
      <variable name="callgroup" value="techsupport"/>
    </variables>
  </user>
</include>
```

然后修改拨号计划(Dialplan)使其它用户可以呼叫到它。
先要修改 conf\dialplan\public.xml 修改为任意号码拨入，
public.xml 里面 public_extensions 修改如下：

```
    <extension name="public_extensions">
      <condition field="destination_number" expression="^([0-9]\d+)$">
     <action application="transfer" data="$1 XML default"/>
      </condition>
    </extension>
```

然后 修改 conf\dialplan\default.xml
在开头部分增加：

```
<extension name="Local_Extension2">
      <condition field="destination_number" expression="^([0-9]\d+)$">
 <action application="export" data="dialed_extension=$1"/>
 <action application="set" data="call_timeout=10"/>
  <action application="set" data="record_sample_rate=8000"/>
  <action application="export" data="RECORD_STEREO=false"/>
 <action application="set" data="hangup_after_bridge=true"/>
 <action application="set" data="continue_on_fail=true"/>
 <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
 <action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
 <action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
      </condition>
    </extension>
```

最后重启 FreeSwitch，就可以使用。

68. FreeSwitch 拨号计划的正则表达式有哪些是最常用的模式？

拨号计划使用 perl 的正则表达式
常用的 匹配 模式如下：
^ 表示开始匹配，^123 表示匹配 123 开头
$表示结束匹配 456$表示匹配 456 结束
| 表示或者，匹配任何一个
[] 表示匹配其中的任意一个字符
[0-9] 等于匹配 [0123456789]
\d 等于 匹配[0-9]
\d+ 等于匹配 1 个或多个数字
\d* 等于匹配 0 个或多个前面的字符

69. FreeSwitch 默认配置如何修改拨号计划设置没有注册上来不走留言信箱？

系统默认的配置呼叫某个分机假如不通会进入留言信箱，出来一段英文的提示，
实际使用中比较恶心。实际使用通常需要去掉这个提示。
修改/conf/dialplan 目录下的 default.xml 文件
在 <context name="default"> 之后加入：

```
<extension name="Local_Extension2">
    <condition field="destination_number" expression="^(1[01][0-9][0-9])$">
  <action application="export" data="dialed_extension=$1"/>
  <action application="set" data="call_timeout=10"/>
  <action application="set" data="hangup_after_bridge=true"/>
  <action application="set" data="continue_on_fail=false"/>
  <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
    </condition>
  </extension>
```

其中的
destination_number 是分机号码
这个例子表示分机号码是 1000-1199 共 200 个

70. FreeSwitch 默认配置加载哪些编码器？

使用 show codec 命令可以看到系统加载的编码器

```
FreeSwitch> show codec
type,name,ikey
codec,ADPCM (IMA),mod_spandsp
codec,AMR,mod_amr
codec,G.711 alaw,CORE_PCM_MODULE
codec,G.711 ulaw,CORE_PCM_MODULE
codec,G.722,mod_spandsp
codec,G.723.1 6.3k,mod_g723_1
codec,G.726 16k,mod_spandsp
codec,G.726 16k (AAL2),mod_spandsp
codec,G.726 24k,mod_spandsp
codec,G.726 24k (AAL2),mod_spandsp
codec,G.726 32k,mod_spandsp
codec,G.726 32k (AAL2),mod_spandsp
codec,G.726 40k,mod_spandsp
codec,G.726 40k (AAL2),mod_spandsp
codec,G.729,mod_g729
codec,GSM,mod_spandsp
codec,H.261 Video (passthru),mod_h26x
codec,H.263 Video (passthru),mod_h26x
codec,H.263+ Video (passthru),mod_h26x
codec,H.263++ Video (passthru),mod_h26x
codec,H.264 Video (passthru),mod_h26x
codec,LPC-10,mod_spandsp
codec,PROXY PASS-THROUGH,CORE_PCM_MODULE
codec,PROXY VIDEO PASS-THROUGH,CORE_PCM_MODULE
codec,RAW Signed Linear (16 bit),CORE_PCM_MODULE
codec,Speex,mod_speex
26 total.
```

但是要注意，有加载不一定就能使用。

具体能否使用，使用哪个还要看 vars.xml 里面的配置

71. FreeSwitch 默认配置哪些编码器能使用？

上面说到系统加载了很多编码器，但是不一定能使用，具体能使用哪些编码器，要看 conf/ vars.xml 配置文件里面的下面的参数：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM"/>
```

global_codec_preFreeSwitch 是全局能使用的编码器。

outbound_codec_preFreeSwitch 是 FreeSwitch 拨出的时候使用的编码器

其中 G722 比较少使用，通常可以去掉。

PCMU PCMA GSM 都是最常用的编码器。

72. FreeSwitch 如何设置修改默认配置添加支持 G.729，iLBC 等编码器？

首先 查看：conf\autoload_configs 目录下的 modules.conf.xml 配置文件

看看是否有打开注释加载到系统，注释了就是没有加载的。

比如 mod_ilbc 默认是没有加载的，而 mod_g729 默认是加载的。因此将

<!--<load module="mod_ilbc"/>-->

改为：

<load module="mod_ilbc"/>

然后在

conf/ vars.xml 配置文件里面的看下面的参数：

<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=G722,PCMU,PCMA,GSM"/>

<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM"/>

假如没有就加上：

<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=PCMU,PCMA,GSM,G729,iLBC"/>

<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM,G729,iLBC"/>

然后保存。重启 FreeSwitch 就可以。

其它说明：

有人说我加了 G729，但是还是不行，那是因为官网的 FreeSwitch 的 G729 只能支持透传的方式，不能转码。

导致呼叫到 IVR，或者两个软电话一个是 G729 一个不是 G729 也不能通话。

要想使用 G729 通话，只能是两个软电话都是 G729 的情况才行，也就是 pass through 透传的方式。

所以不建议使用 G729，但是由于很多落地都是使用 G729，那么只能是要求软电话首选 G729 拨入。

假如要支持 G729 转码的请参见如何支持 G729 转码的问题。

73. 如何看 FreeSwitch 当前支持哪些语音和视频编码器？

结合使用 show codec 命令可以看到系统加载的编码器

和 sofia status profile internal 或者 5080 端口的 sofia status profile external

比如我机器上，显示如下下面斜体的部分就是说明了只有支持 ALAW

freeswitch@yhy-PC32> sofia status profile internal

==========

Name internal

Domain Name N/A

Auto-NAT false

DBName sofia_reg_internal

Pres Hosts 192.168.1.102,192.168.1.102

Dialplan XML

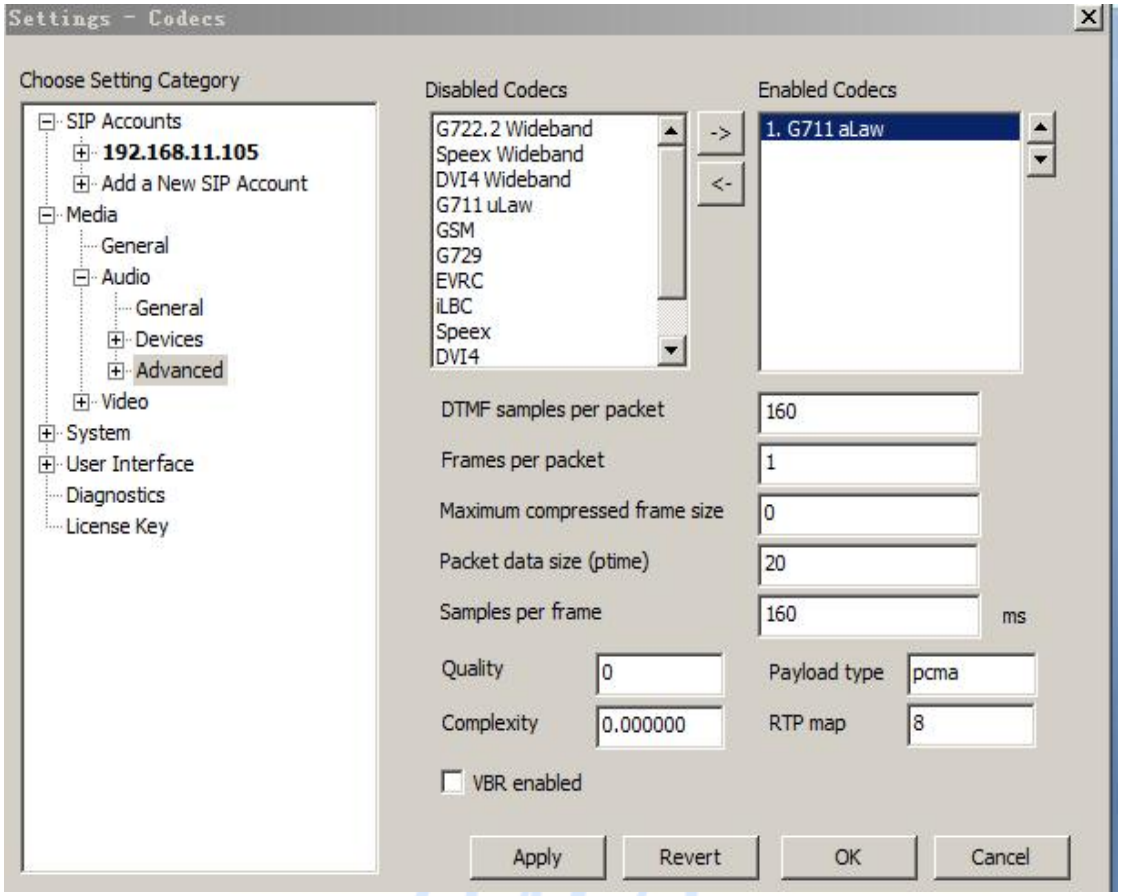
Context public

Challenge Realm auto_from

RTP-IP	192.168.1.102
SIP-IP	192.168.1.102
URL	sip:mod_sofia@192.168.1.102:5060
BIND-URL	sip:mod_sofia@192.168.1.102:5060
HOLD-MUSIC	local_stream://moh
OUTBOUND-PROXY	N/A
CODECS IN	*PCMA*
CODECS OUT	*PCMA*
TEL-EVENT	101
DTMF-MODE	rfc2833
CNG	13
SESSION-TO	18000
MAX-DIALOG	10000
NOMEDIA	false
LATE-NEG	false
PROXY-MEDIA	false
ZRTP-PASSTHRU	false
AGGRESSIVENAT	false
STUN-ENABLED	true
STUN-AUTO-DISABLE	false
CALLS-IN	0
FAILED-CALLS-IN	0
CALLS-OUT	3
FAILED-CALLS-OUT	2
REGISTRATIONS	0

74. 软电话上如何指定编码器

大多数软电话上都能指定使用编码器
我们强烈建议在内网测试，首先只要指定 G.711 alaw 语音编码器就可以。
简单明了，有问题也好查，因为 SDP 的协商过程太过复杂，初学者很晕。
以 eyebeam 为例子，在 eyebeam 上点右键，点 setting 然后 点 media，audio，advance 如下图：



（通常情况下视频是不需要的，可以都去掉，只有当需要视频功能的时候才加上 H.263，H.264）

75. 如何实现让 FreeSwitch 进行转码？

FreeSwitch 的默认配置是不进行转码的：
比如软电话 a 只有 G.711 alaw 编码器，软电话 b 只有 G.711 ulaw 编码器。
它们都注册到 FreeSwitch 上，注册是成功的，
但是无论 a 呼叫 b 还是 b 呼叫 a 都是不行的。
但是 a 呼叫 a 自己是可以的，b 呼叫 b 自己也是可以的。
要让 FreeSwitch 支持分机间能转码通话需要修改：
conf\sip_profiles\ 目录下的 internal.xml 或 external.xml（5080 端口上的）下面参数设置为 false：

```
<param name="inbound-zrtp-passthru" value="false"/>
```

要提高系统效率就需要尽量不转码，也就是说尽量 b 使用 a 提供的编码器，配置：

```
<param name="inbound-late-negotiation" value="true"/>
<param name="disable-transcoding" value="true"/>
```

然后在拨号计划里面设置下面的参数来优先使用拨入的软电话的编码器，以减小系统转码的概率：

```
<action application="set" data="inherit_codec=true"/>
```

76. FreeSwitch 哪些编码器不支持转码？

默认情况下在许多需要专利付费的编码器，fs 都是不支持转码，只能透传。

FreeSwitch 里面的 G.729 ，G.723.1 和 AMR 等都是*只能透传*的。
无法实现转码也无法实现到 ESL 的 IVR 播音啥的。G729 要实现转码需要修改源码。

77. 如何才能让 FreeSwitch 支持 G729 转码？

要实现 G.729 的转码理论上需要自己开发编码器，即在 FreeSwitch 的源码上进行修改。
幸好有人已经基于 ITU-T G.729A 开源的代码上帮我们做了这个事情，改动之后的源代码打包叫做：mod_g729.zip。mod_g729.zip 已经放在 42.120.20.89 上，目前仅仅支持 linux 版本。
首先要保证 FreeSwitch 的源码编译通过，可以使用。
然后下载 mod_g729.zip 下载之后，拷贝到 linux 下，比如在 /下，然后解压：
unzip mod_g729.zip
然后删除原来的 mod_g729 的源码模块,假设 FreeSwitch 的源码在/fs
cd /fs/src/mod/codecs
rm –rf mod_g729
然后移动新的源码到 codec 目录中
cd /
mv mod_g729 /fs/src/mod/codecs/
然后编译 mod_g729
cd /fs/src/mod/codecs/mod_g729
make
然后重新编译和安装 FreeSwitch：cd /fs
make
make install
然后 在 conf/ vars.xml 配置文件里面加上 G729 编码支持：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=PCMU,PCMA,GSM,G729,iLBC"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM,G729,iLBC"/>
```

然后保存，启动 FreeSwitch，使用两个客户端：一个只有 alaw，另外一个只有 G729 就可以测试转码。
也可以使用 ESL 的 IVR 进行测试。以上 在 centos 6.3 操作系统和 freeswitch-1.2.5.3 的版本下测试通过，包括 ESL 的 IVR 播音和取按键在内的功能都可以完美支持。这个压缩包是基于 ITU-T G.729A 开源的代码上实现的，作为功能测试小规模不会有问题，作为实网应用的时候，大规模并发性能上不去，另外也存在两个通道录音不同步的问题。本人另外实现了基于 intel 高性能科学计算库 IPP 的 G729 转码，并发转码性能大幅提高也解决录音不同步的问题，在 E5-2640 双 CPU 服务器上可以支持 1000 线转码，有需要可以联系本人。

78. FreeSwitch 如何设置修改默认配置才能支持视频通话？

在 conf/ vars.xml 配置文件里面修改下面的参数，增加 H263 或者 H264 编码：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM"/>
```

改为：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=PCMU,PCMA,GSM,H263"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM,H263"/>
```

然后保存。重启 FreeSwitch 就可以。
要测试支持视频通话，软电话要支持。

Windows 下个人建议使用 MicroSip 软电话测试视频。

其它说明：FreeSwitch 的视频通话是不支持转码的，因此使用的时候需要所有的软电话都设置为一样的视频编码器，比如都设置为 H263 编码器

LAN 局域网环境下测试：

使用 android 下的两个 ImsDroid 测试，conf/ vars.xml 配置文件修改为：

```
<X-PRE-PROCESS cmd="set"
data="global_codec_preFreeSwitch=PCMU,PCMA,GSM,H263,H263-1998,H263-2000,H264"/>
  <X-PRE-PROCESS cmd="set"
data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM,H263,H263-1998,H263-2000,H264"/>
```

然后在两个手机上启动 ImsDroid 进行测试。ImsDroid 设置一样的语音编码器和一样的视频编码器然后呼叫测试。可以进行视频通话。

假如设置不一样的视频编码器，可以通话，但是无法看见视频的，这个也说明了 FreeSwitch 不进行视频转码而只是进行视频转发。

ImsDroid 本人测试过以下视频编码：

H.263 Video
H.263-1998 Video （h263+）
H.263-2000 Video （h263++ ）
H.264 Video Base Profile
H.264 Video Main Profile

WAN 广域网环境下测试：

两个客户端都使用 android 下的 ImsDroid 测试，视频呼叫不能建立，不过语音呼叫可以建立，原因未明。

后面改为使用 android 下的两个 Linphone2.0.2 测试，都选择一样的视频编码器（比如 H.264）和语音编码器，可以进行视频通话。

79. FreeSwitch 如何设置修改默认配置才能支持使用 VP8 编码进行视频通话？

VP8 是 Google 开源的最新视频编码器，基于 H.264 改进，至于改进的真正效果，众说纷纭。

VP8 是如此重要因为他是 Google 搞的，是 WebRTC 默认的视频编码器，你要使用 WebRTC 假如有视频，就需要支持 VP8.

关于 VP8 视频编码：

FreeSwitch windows 下 vp8 没有工程文件，因此没有产生 mod_vp8.dll,

不支持 VP8 编码进行视频通话。本人只好自己手工打造了一个 mod_vp8.dll,

有需要的伙计可以联系本人索取。

Linux 版本 编译，默认就会产生 mod_vp8.so.所以可以测试。

conf/ vars.xml 配置文件修改为：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=PCMU,PCMA,GSM,VP8"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=PCMU,PCMA,GSM,VP8"/>
```

然后使用 ImsDroid 支持 VP8 编码的版本进行测试，语音编码和视频编码都设置一样的编码器，ImsDroid 视频编码都只选择 VP8 编码。

然后呼叫测试，可以进行视频通话，对 FreeSwitch 来说 VP8 编码也是透传。

80. FreeSwitch 如何在服务端指定注册周期？

FreeSwtich 提供参数在服务端指定注册周期，比如服务器可以指定客户端用户每隔 1800 秒注册一次，而无论客户端用户设置多少秒都是无效的,即使客户端上设置 10 秒，或者 10000 秒，都无效，注册周期都是 1800 秒。

在 \conf\sip_profiles\目录下 internal.xml 或者 external.xml

里面添加：

<param name="sip-force-expires" value="1800"/>

根据测试在 sofia.conf.xml 配置无效。

还有一个参数是，用来防止注册风暴。

比如：

<param name="sip-expires-max-deviation" value="100"/>

表示

从 sip-force-expires -100 到 sip-force-expires+100 之间的秒数。随机产生一个时间给某个客户端，防止所有的客户端注册周期一样导致大家在统一时间一起注册，从而引起注册风暴。

上面的参数可以在 用户字典里面针对某个用户进行指定。

81. FreeSwitch 如何在拨号计划里面设置时间条件？

以小时为单位：假设 9:00-18:59

```
<extension name="Local_Extension0" continue="true">
<condition hour="9-18">
        <action application="set" data="call_timeout=10"/>
        <action application="set" data="sofia_session_timeout=130"/>
        <action application="export" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=false"/>
        <action application="sched_hangup" data="+10 alloted_timeout"/>
    <action application="bridge" data="user/${destination_number}@${domain_name}"/>
</condition>
```

以分钟为单位： minute-of-day = 1-1440

9:00-18:00

```
<extension name="Local_Extension0" continue="true">
<condition minute-of-day="540-1080">
        <action application="set" data="call_timeout=10"/>
        <action application="set" data="sofia_session_timeout=130"/>
        <action application="export" data="hangup_after_bridge=true"/>
```

```
                <action application="set" data="continue_on_fail=false"/>
                <action application="sched_hangup" data="+10 alloted_timeout"/>
            <action application="bridge" data="user/${destination_number}@${domain_name}"/>
        </condition>
```

82. FreeSwitch 如何限制通话时间，如何定时挂机？

通过设置 sip_profiles\internal.xml

```
<param name="enable-timer" value="true"/>
<param name="minimum-session-expires" value="120"/>
<param name="session-timeout" value="1800"/>
```

发现无效。

于是使用调度的功能来实现：

假设 a 转 b，60 秒之后挂机：

a 转 b，60 秒之后挂机，包括 b 的振铃时间：

```
<action application="sched_hangup" data="+60 alloted_timeout"/>
<action application="bridge" data="user/${destination_number}@${domain_name}"/>
```

a 转 b，60 秒之后挂机，从 b 接通开始计算：

```
<action application="set" data="execute_on_answer=sched_hangup +60 alloted_timeout" />
<action application="bridge" data="user/${destination_number}@${domain_name}"/>
```

a 转 b，60 秒之后挂机 a 先进入 ivr，然后转 b，接通 60 秒之后挂机：

```
<action application="export" data="nolocal:api_on_answer=sched_hangup +60 ${uuid} alloted_timeout" />
<action application="bridge" data="user/${destination_number}@${domain_name}"/>
```

第四章 FreeSwitch 高级配置部分

83. FreeSwitch 如何实现在两个网卡上不同的网段上的分机互通？

在两个网卡上 不同的 网段上,与运行两个网段上的软电话互通.
这是一个实际应用环境中会经常碰到的问题
比如： 192.168.1.X 和 192.168.11.X 网段
1000 分机 register on internal 192.168.1.X 5060
1002 分机 register on external 192.168.11.X 5060
拨打测试：
external 1002 分机呼叫 internal 1000 分机成功
internal 1000 分机 呼叫 external 1002 分机失败
需要修改文件 internal.xml 和 external.xml
位置： conf/sip_profiles/目录下
internal.xml 修改内容：

```
<param name="rtp-ip" value="192.168.1.101"/>
<param name="sip-ip" value="192.168.1.101"/>
<param name="sip-port" value="5060"/>
<param name="inbound-late-negotiation" value="false"/>
<param name="inbound-zrtp-passthru" value="false"/>
```

external.xml

```
<param name="rtp-ip" value="192.168.11.101"/>
<param name="sip-ip" value="192.168.11.101"/>
<param name="sip-port" value="5060"/>
<param name="inbound-late-negotiation" value="false"/>
<param name="inbound-zrtp-passthru" value="false"/>
```

打开 sip_profile/internal.xml 文件，反注释相同的行：

```
<param name="force-register-domain" value="$${domain}"/>
<param name="force-register-db-domain" value="$${domain}"/>
<param name="dbname" value="share_presence"/>
<param name="presence-hosts" value="$${domain}"/>
```

打开 sip_profile/external.xml 文件，反注释下面的行：

```
<param name="force-register-domain" value="$${domain}"/>
<param name="force-register-db-domain" value="$${domain}"/>
<param name="dbname" value="share_presence"/>
<param name="presence-hosts" value="$${domain}"/>
```

默认的拨号计划，只能呼叫注册在 192.168.1.X 网段上的 internal 上的分机。
要呼叫 192.168.11.X 上的号码，需要通过 gateway 的方式设置拨打，或者按照下面的方法修改拨

号计划：

```
<extension name="Local_Extension2">
    <condition field="destination_number" expression="^([0-9]\d+)$">
  <action application="export" data="dialed_extension=$1"/>
  <action application="set" data="call_timeout=30"/>
   <action application="set" data="record_sample_rate=8000"/>
   <action application="export" data="RECORD_STEREO=false"/>
  <action application="set" data="hangup_after_bridge=false"/>
  <action application="set" data="continue_on_fail=true"/>
      <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
      <action application="bridge" data="sofia/external/${dialed_extension}%192.168.1.101"/>
   </condition>
  </extension>
```

说明：上面的斜体部分其中的 192.168.1.101 是内网的网卡 ip 地址（而不是外网的 IP 192.168.11.101），这个很关键。或者可以使用${domain_name} 代替，不过要在 vars.xml 里面先指定：

```
<X-PRE-PROCESS cmd="set" data="domain=192.168.1.101"/>
```

84. FreeSwitch 如何支持 3 个 IP 或 profile 上的分机互通？

实际应用环境中还可能出现需要 3 个 profile 的情况，比如机器上有 3 个 IP 地址：一个内网地址，两个外网地址，接不同的运营商。这个时候需要 3 个 profile 才能支持。

环境假设：

假设机器上有个 IP 地址和对应的 profile 名称如下：

192.168.1.236:5060 internal

192.168.2.236 :5060 internal_yhy

10.10.10.236:5060 external

为了测试方便，统一使用 5060 作为 sip 端口。

配置步骤如下：

首先，拷贝 conf\sip_profiles\internal.xml 文件一份，新文件重命名为 internal_yhy.xml

这样 conf\sip_profiles 下应该有 3 个 profiles，如下：

internal.xml ,internal_yhy.xml 和 external.xml

然后，打开 internal_yhy.xml 修改 <profile name="internal_yhy">

接着，参照上面的问题，修改 3 个 profile 里面的下面参数（假如没有就手工添加）：

```
<param name="force-register-domain" value="$${domain}"/>
  <param name="force-register-db-domain" value="$${domain}"/>
  <param name="dbname" value="share_presence"/>
  <param name="presence-hosts" value="$${domain}"/>
```

然后，修改 vars.xml 里面的参数：

```
<X-PRE-PROCESS cmd="set" data="domain=192.168.1.236"/>
  <X-PRE-PROCESS cmd="set" data="internal_sip_port=5060"/>
  <X-PRE-PROCESS cmd="set" data="internal_tls_port=5061"/>
  <X-PRE-PROCESS cmd="set" data="external_sip_port=5060"/>
  <X-PRE-PROCESS cmd="set" data="external_tls_port=5061"/>
```

修改 internal.xml 里面的参数：

```
<param name="rtp-ip" value="192.168.1.236"/>
<param name="sip-ip" value="192.168.1.236"/>
```

修改 internal_yhy.xml 里面的参数：

```
<param name="rtp-ip" value="192.168.2.236"/>
<param name="sip-ip" value="192.168.2.236"/>
```

修改 external.xml 里面的参数：

```
<param name="rtp-ip" value="10.10.10.236"/>
<param name="sip-ip" value="10.10.10.236"/>
```

最后修改拨号计划如下：

```
<extension name="Local_Extension2">
    <condition field="destination_number" expression="^([0-9]\d+)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="set" data="call_timeout=30"/>
        <action application="set" data="record_sample_rate=8000"/>
        <action application="export" data="RECORD_STEREO=false"/>
        <action application="set" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=true"/>
        <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
<action application="bridge" data="sofia/internal_yhy/${dialed_extension}%${domain_name}"/>
        <action application="bridge" data="sofia/external/${dialed_extension}%${domain_name}"/>
        </condition>
</extension>
```

万事大吉，找三个软电话分别注册到上面的三个 IP 地址：192.168.1.236，192.168.2.236 和 10.10.10.236 上，然后开始测试，先自己拨打自己，可以了再互相拨打。

85. 没有注册的软电话如何实现对 FreeSwitch 分机的呼叫？

办法 1.修改：acl.conf.xml

比如允许 192.168.11.X 网段拨入到软电话或者 flash phone，acl.conf.xml 修改如下，然后没有注册的呼叫注册的,可以直接呼到默认的配置 的 ip 的 5060 或者 5080 端口就行

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <!--
    These ACL's are automatically created on startup.
    rfc1918.auto    - RFC1918 Space
    nat.auto        - RFC1918 Excluding your local lan.
    localnet.auto - ACL for your local lan.
    loopback.auto - ACL for your local lan.
     -->
     <list name="lan" default="allow">
        <node type="deny" cidr="192.168.42.0/24"/>
        <node type="allow" cidr="192.168.42.42/32"/>
```

```
    </list>
    <!--
   This will traverse the directory adding all users
   with the cidr= tag to this ACL, when this ACL matches
   the users variables and params apply as if they
   digest authenticated.
    -->
    <list name="domains" default=" allow ">
      <!-- domain= is special it scans the domain from the directory to build the ACL -->
      <node type="allow" domain="$${domain}"/>
      <node type="allow" host="192.168.11.0" mask="255.255.255.0"/>
      <!-- use cidr= if you wish to allow ip ranges to this domains acl. -->
      <!-- <node type="allow" cidr="192.168.0.0/24"/> -->
    </list>
  </network-lists>
</configuration>
```

注意上面的斜体部分。默认都改为允许就可以。

办法 2.修改：vars.xml

```
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
  <X-PRE-PROCESS cmd="set" data="internal_auth_calls=true"/>
```

都改为：

```
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
  <X-PRE-PROCESS cmd="set" data="internal_auth_calls=false"/>
```

然后 5060 端口就跟 5080 端口一样，不用注册就可以随便拨打。

不过这两个做法是有巨大风险的，实网投产系统绝对不能这么做，否则总有一天会因为会被人盗打而吃大亏。

公网上有很多国外的服务器一直扫描 5060 端口，然后乱发瞎蒙被叫，下面是一个实际的日志的被叫号码：

```
called=0010448708757984
called=0020448708757984
called=0014448708757984
called=8972599953205
called=9972599953205
called=100972599953205
called=00972599953205
called=000972599953205
called=900972599953205
called=980097259995320
called=801097259995320
called=8972599953205
called=9972599953205
called=100972599953205
called=00972599953205
called=000972599953205
```

一旦蒙通过，你就惨了。

86. FreeSwitch 为啥会没有发挂机信号给 A leg？

有两种情况会导致这个问题：

A. <action application="set" data="hangup_after_bridge=true"/> 设置成了 false

B. Invite 里面的 from 地址为：1005@FreeSwitch ip 地址发起的会导致被叫挂机,FreeSwitch 不会转发挂机信号给主叫。

因此跟第三方集成的时候，from 的地址要保证是对的。

这种情况可能会导致话路没有挂机，因此一般情况下 hangup_after_bridge=true 这个要设置。

87. FreeSwitch 如何实现多个软电话或者 IP 话机共振？

共振有两种情况：

一种情况是 多个不同的电话机 或者 ip 话机或者软电话 使用同一个分机号注册，然后拨打这个分机，大家一起振铃。

FreeSwitch 要实现共振需要修改

conf\sip_profiles 目录下的 internal.xml

```
<!--<param name="multiple-registrations" value="contact"/> -->
```

改为：

```
<param name="multiple-registrations" value="true"/>
```

然后重启 FreeSwitch，使用不同的软件 相同的分机号码注册 就可以实现共振。

假如不想设置共振，改为 contact 或者设置为 false。

参数说明：contact 与 false 的区别在于：

设置为 contact 的时候，是根据 sip_user, sip_host and contact 字段判断的，

设置为 false 的时候是基于 Call-ID 判断。

因此假如是公网运营，设置为 contact 的时候可能会导致一个软电话刚刚启动的时候多个帐号注册上来（实际上就一个帐号），设置为 false 就不会。

因此建议要么设置为 true，要么设置为 false，不要设置为 contact，因为会导致莫名其妙的问题。

实现共振的分机号码 IM 也是支持多个显示的，比如 1002 分机共振，那么发给 1002 的 IM 消息所有注册上来的终端都会收到 IM 的消息。

其它说明:

根据 http://wiki.FreeSwitch.org/wiki/Mod_sofia 介绍

需要拨号计划拨打的时候还需要：

```
<action application="bridge" data="${sofia_contact(${dialed_extension})}"/>
```

才能共振，但是实际测试。FreeSwitch 1.2.7 的版本，默认的拨打分机方式：

```
<action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
```

一样可以共振。应该是参数= contact 的时候加上上面的拨号计划的修改实现共振。

几种不同的组合，根据你的实际情况看看需要采用哪种。

另外一种情况是多个分机号码组成一组，然后拨打里面的某个号码，这组的所有分机号码都会振铃。这种情况的实现可以有两种办法。

A）是通过组呼叫来实现：

先在 conf\directory 目录下的 default.xml 文件 定义一个组，比如下面的 salse 组：

```
<group name="sales">
        <users>
          <!--
              type="pointer" is a pointer so you can have the
              same user in multiple groups.   It basically means
              to keep searching for the user in the directory.
          -->
          <user id="1000" type="pointer"/>
          <user id="1001" type="pointer"/>
          <user id="1002" type="pointer"/>
          <user id="1003" type="pointer"/>
          <user id="1004" type="pointer"/>
        </users>
    </group>
```

然后在 conf\dialplan 目录下的 default.xml 拨号计划里面加上：

```
<extension name="group_dial_sales">
      <condition field="destination_number" expression="^2000$">
        <action application="bridge" data="${group_call(sales@${domain_name})}"/>
      </condition>
</extension>
```

这样用户拨打的时候整个组的人都会被呼叫。

b）是通过 ESL 自己编程来实现，步骤如下：

拨入到 esl 之后，先应答拨入的会话，然后查询数据库，找出所有共振的号码(同组的号码)，假如只有一个号码，或者说没有共振，直接进行 bridge 操作。

假如有多个共振号码。先给拨入的会话播放振铃声音，然后对所有的共振号码使用 originate 发起呼叫，

一旦有人接通，把接通的分机号码跟拨入的会话进行 bridge，然后挂掉所有其它正在振铃的号码。这个过程比较麻烦，但是很灵活。

具体如何使用 esl 开发，请参见关于 FreeSwitch ESL 编程部分。

88. FreeSwitch 如何修改默认配置才能拨打外部的 SIP 电话或者 SIP 网关？

在 conf\sip_profiles\external 目录下建立：

gw1.xml 内容如下：

```
<gateway name="gw1">
    <param name="realm" value="192.168.11.103:5060"/>
    <param name="username" value="5678"/>
```

```
        <param name="password" value="1234"/>
        <param name="register" value="false" />
        <param name="caller-id-in-from" value="true"/>
    </gateway>
```

其中 ip 地址和端口是外拨网关的 ip 和 sip 的端口,注意上面的斜体 caller-id-in-from 参数表示使用拨入的主叫（比如用户的注册分机号码）作为拨打出去的真正主叫。假如没有设置，那么拨打网关出去显示的主叫是固定的，主叫都是上面的 5678。

然后 有两种办法可以使用这个网关：

办法 1:

在 conf\dialplan\default 建立一个 call_out.xml

内容如下：

```
<include>
  <extension name="call out">
    <condition field="destination_number" expression="^9(\d+)$">
      <action application="bridge" data="sofia/gateway/gw1/$1"/>
    </condition>
  </extension>
</include>
```

表示拨打 9 开头的号码都走外拨网关 gw1 的路由

办法 2:

也可以在 conf\dialplan\ default.xml 拨号计划里面直接使用：

```
<extension name="Local_Extension2">
    <condition field="destination_number" expression="^9(\d+)$">
            <action application="export" data="dialed_extension=$1"/>
            <action application="answer" />
            <action application="set" data="call_timeout=30"/>
             <action application="set" data="record_sample_rate=8000"/>
             <action application="export" data="RECORD_STEREO=false"/>
            <action application="set" data="hangup_after_bridge=true"/>
            <action application="set" data="continue_on_fail=true"/>
            <action application="bridge" data="sofia/gateway/gw1/$1"/>
        </condition>
</extension>
```

个人建议直接在拨号计划里面使用比较直观。

89. 外部的 SIP 网关如何拨打到某个分机？

在 conf/dialplan/public 目录下建立 my_did.xml 文件内容如下：

```
<include>
  <extension name="public_did">
    <condition field="destination_number" expression="^(你的接入号码)$">
      <action application="transfer" data="1000 XML default"/>
    </condition>
```

```
    </extension>
</include>
```

之后拨打你的接入号码就会到分机上。

还有其它办法，比如外部的网关以分机注册到 FreeSwitch 上，当然也可以拨打分机。

90. FreeSwitch 如何和迅时网关 MX8 进行集成？

假设：

FreeSwitch 的 ip 是 192.168.1.236

mx8 的 ip 是 192.168.1.251

首先解决用户通过 MX8 拨入到 FreeSwitch 上的问题：

配置 MX8，使它以分机的形式注册到 FreeSwitch 上。

有用户拨入的时候指向外联模式设置的接入号码，从而进入 ivr 系统。

主要配置界面如下：

Mx8 ip 配置界面：

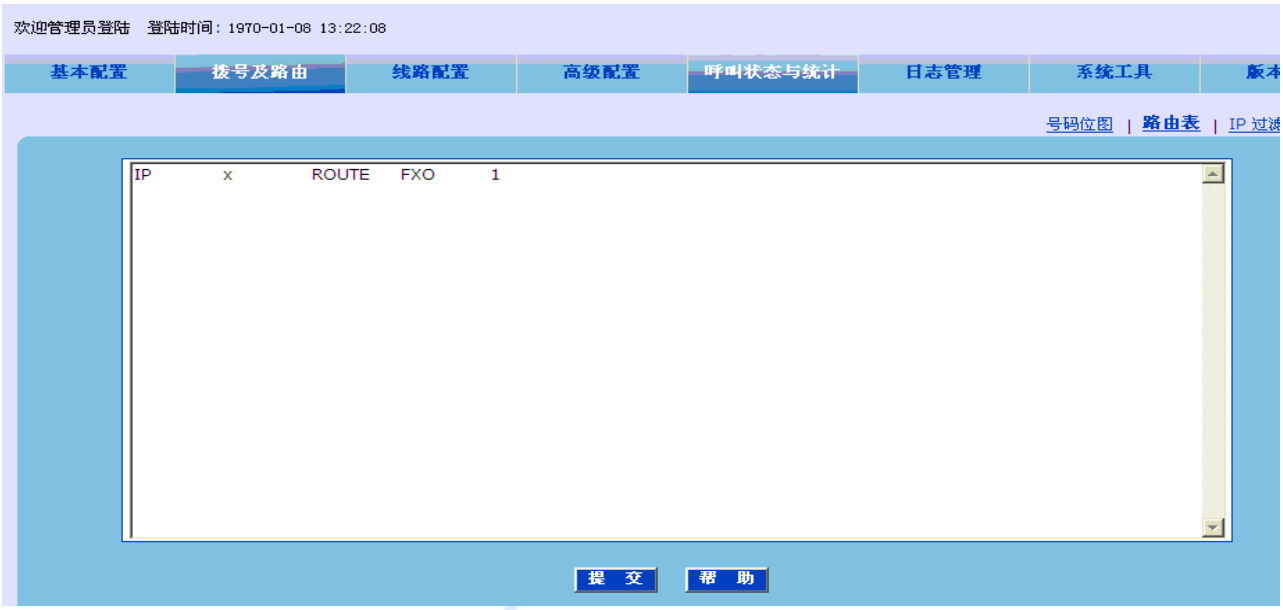
MX8 系统配置界面：

MX8 SIP 配置界面：

192.168.1.236 是 FreeSwitch 的 ip

注册用户名是 1005 默认密码是 1234

MX8 拨出路由配置界面：默认所以的 ip 拨入都从 PSTN 第一线 拨出：



MX8 线路号码配置界面：

MX8 线路绑定号码配置界面：

其中的 123961 是 ivr 系统的接入号码，

假如电话线路有开反极性检测 可以勾选反极性检测，假如没有开不能勾选。

其次解决拨出的问题：通常 mx8 是作为网关出口，因此创建 conf\sip_profiles\external\gw1.xml 内容如下：

```
<gateway name="gw1">
    <param name="realm" value="192.168.1.251:5060"/>
    <param name="username" value="5678"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
 </gateway>
```

然后在需要拨打外线的时候，

分为两种情况

1） 分机上外拨：

从某一分机上呼出：打外部电话就需要加先拨 9

修改拨号计划，在 conf\dialplan\ default.xml 拨号计划里面：

```
<extension name="Local_Extension2">
  <condition field="destination_number" expression="^9(\d+)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="answer" />
        <action application="set" data="call_timeout=30"/>
         <action application="set" data="record_sample_rate=8000"/>
         <action application="export" data="RECORD_STEREO=false"/>
        <action application="set" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=true"/>
        <action application="bridge" data="sofia/gateway/gw1/$1"/>
    </condition>
  </extension>
```

2） ESL 编程上的 ivr 外拨：

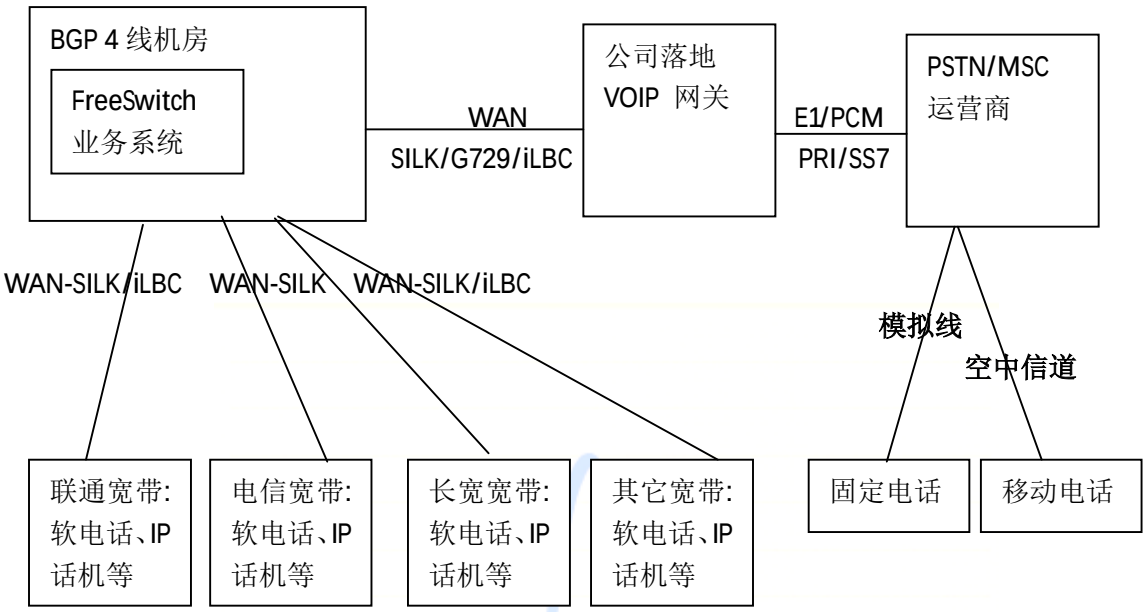
可以判断被叫号码，比如:被叫号码超过 4 位的时候，作为外线，或者 9 开头的号码作为外线当是外线的时候，执行以下 C 函数：

```
sprintf(cmd_tmp,"bgapi originate
{origination_uuid=%s,originate_timeout=60}sofia/gateway/gw1/%s %s\n\n",uuid,called,system_caller);
 res=esl_send_recv(&handle,cmd_tmp);
```

91. FreeSwitch 公网运营如何设计？

FreeSwitch 运行在公网，意味着：

A. 假如是对公众提供服务，软电话分机分布在全国各地，各个运营商都可能存在，因此各个运营商的接入带宽延时是要考虑的。我们建议部署在 BGP 4 线机房
B. 意味这软电话在公司内网，需要 NAT 穿透才能使用。
C. FreeSwitch 公网运行的一种建议方案如下图：



92. FreeSwitch 公网服务在路由器防火墙之后如何部署？

部署在防火墙之后意味着 FreeSwitch 运行的机器上是没有公网 IP 的，只有内网 IP。
公网 IP 是通过 IP 映射或者端口映射到 FreeSwitch 的机器上。
理论上启动 FreeSwitch 的时候不带任何参数自动支持 NAT，但是本人测试这个方法不靠谱，尤其麻烦的是不稳定，有点象段公子的六脉神剑，经常不灵光。
不稳定的结果是经常导致一方单通，甚至双方都没有声音的情况。
比较保险的做法是修改以下的配置文件：conf\sip_profiles\internal.xml 里面的：

```
<param name="ext-rtp-ip" value="auto-nat"/>
<param name="ext-sip-ip" value="auto-nat"/>
```

直接修改为公网地址，比如 218.107.217.204 是外网地址，修改如下：

```
<param name="ext-rtp-ip" value="218.107.217.204"/>
<param name="ext-sip-ip" value="218.107.217.204"/>
```

对应的 external.xml 配置里面也有这个参数可以根据需求进行修改。
假如是端口映射，那么默认端口需要在路由器上映射。
下面这些默认端口需要映射到 FreeSwitch 的机器上：

5060	UDP & TCP	SIP UAS	Used for SIP signaling (Standard SIP Port, for default Internal Profile)
5080	UDP & TCP	SIP UAS	Used for SIP signaling (For default "External" Profile)
16384-32768	UDP	RTP/ RTCP multimedia streaming	RTP 的语音和视频数据传输用端口，要修改这个范围，请修改 switch.conf.xml 里面的： <param name="rtp-start-port" value="15000"/> 和<param name="rtp-end-port" value="16000"/>参数

说明：配置外网的 IP 之后，假如软电话要内网使用，需要路由器上映射 SIP 端口号是一致的才行。
比如：外网是 5060，映射到内网的 IP 的 5060 端口上。否则有可能会导致通话 30 秒左右就掉线的情况。

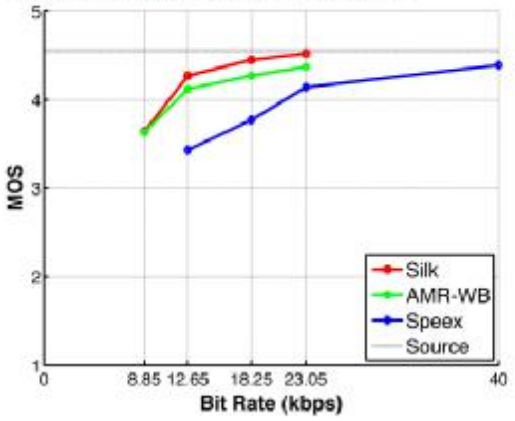
93. FreeSwitch 公网运营有哪些需要特别考虑的？

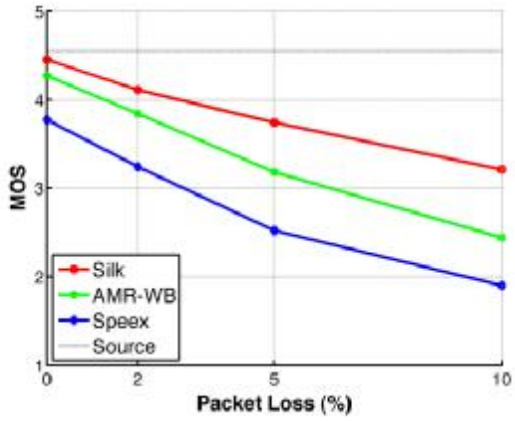
除了考虑实网部署的问题之外
公网运营考虑的问题 还有带宽核算和安全性，
带宽的问题主要是编码器的使用由于 G729 默认是没有转码的。
因此考虑 iLBC 编码是合适的,本人测试公网最好的编码效果是 SILK 编码。
带宽核算需要会精确计算带宽，具体参照前面的带宽问题。

94. FreeSwitch 公网运营如何支持 SILK 编码？

SILK 语音编码是本人测试公网通话效果最好的编码器，而且 FreeSwitch 支持转码。
SILK 编码是来自 SKYPE 的编码器，最近才开源，根据 SKYPE 提供的 datasheet 如下图：

• Highest quality under all conditions[1]





Office Noise, 15 dB SNR	Source	SILK	AMR-WB	Speex
MOS	3.30	3.22	3.14	2.74

• Low Delay: 25 ms (20 ms frame size + 5 ms look-ahead)

SILK 编码器延时 25ms，最重要的是 MOS 声音评分在任何情况下都具有非常高的得分。无论是丢包率多还是少的情况下都比 speex，AMR 之类的要好。
要支持 SILK 编码 FreeSwitch 需要按照如下步骤进行配置：
A. modules.conf.xml 文件里面加上：

```
<load module="mod_silk"/>
```

B. vars.xml 里面加上 SILK 的编码器，比如：

```
<X-PRE-PROCESS cmd="set" data="global_codec_preFreeSwitch=SILK,PCMA,PCMU,GSM,iLBC"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_preFreeSwitch=SILK,PCMA,PCMU,GSM,iLBC"/>
```

C. 找到能支持 SILK 编码器的 sip 软电话或者 sip 话机
本人使用桌面版 pjsip 2.1 版本 和安卓版使用 linphone for android 2.0.2 版本测试是支持的。
其它说明：
有一个地方需要注意的是，SILK 支持多种采样编码如下图：

	Sampling Rate (kHz)	Bit Rate (kbps)	CPU (MHz on x86 core)
Narrowband for PSTN gateways and low end devices	8	6 - 20	12 - 30
Mediumband for devices with limited wideband capability	12	7 - 25	16 - 40
Wideband for all-IP platforms	16	8 - 30	20 - 50
Super-Wideband, a new standard in speech quality	24	12 - 40	30 - 80

FreeSwitch 以上 4 种都支持。
但是，linphone for android 2.0.2 版本 只有支持 16K 的，pjsip 2.1 版本支持 8K 的和 16K 的两种，默认是 8K 的编码优先，因此假如是 linphone for android 2.0.2 呼叫 pjsip 2.1 是没有问题的，但是假如是 pjsip 2.1 呼叫 linphone for android 2.0.2 会有问题。
除非你修改 pjsip 2.1，让 16K 的 SILK 编码器优先，这样，互相呼叫都不会有问题。

另外 pjsip 2.1 版要支持 SILK 需要稍微修改折腾一下 silk 库（默认的安装包里面不带 SILK 源码，需要从 FreeSwitch 里面搞出来，一些参数库名称也需要修改）。
pjsip 2.1 增加 SILK 编码器有需要&&有困难的伙计可以 mail || qq 联系我。

95. FreeSwitch 公网运营环境下哪些情况下测试过？

联通 WCDMA 3G 数据通道。
电信 CDMA 2000 3G 数据通道。
电信宽带用户。
联通宽带用户。
长宽宽带用户。
移动（铁通）宽带用户。
客户软电话在公司内部通过 WIFI 上网的环境测试
移动的 4G TD-LTE，这在窗口的时候带宽可以到 60M，在室内大约 10-20M。

96. FreeSwitch 如何禁止 IP 改变或者网线接触不好情况下自动重启 sofia 模块？

修改文件 /conf/autoload_configs/sofia.conf.xml，修改内容：

```
<param name="auto-restart" value="false"/>
```

该属性设置的目的是防止 FreeSwitch 在检测到 IP 地址发生改变或者网络线接触不好或者网络线松掉后等意外情况出现的时候，FreeSwitch 自动重启 sofia 模块。
实际假如没有配置，自动分配 IP 的情况下,本人测试分机通话可能会莫名其妙就掉线（可能是偶然情况，但是有出现一次错误也是错误），日志如下：

```
[INFO] mod_sofia.c:5478 EVENT_TRAP: IP change detected
```

[INFO] mod_sofia.c:5479 IP change detected

[]->[] [2001:0:4137:9e76:20d6:a9b9:c5e9:6889]->[2001:0:4137:9e76:24d7:fa44:c5e9:6889]

[NOTICE] sofia_glue.c:5679 Reload XML [Success]

[INFO] mod_enum.c:872 ENUM Reloaded

[INFO] switch_time.c:1173 Timezone reloaded 530 definitions

[DEBUG] switch_channel.c:3011 (sofia/internal/1001@192.168.1.101) Callstate Change ACTIVE -> HANGUP

[DEBUG] switch_channel.c:3011 (sofia/internal/sip:1007@192.168.1.107:5060) Callstate Change ACTIVE -> HANGUP

[NOTICE] mod_sofia.c:1065 Hangup sofia/internal/sip:1007@192.168.1.107:5060 [CS_EXCHANGE_MEDIA] [NORMAL_CLEARING]

[NOTICE] mod_sofia.c:1065 Hangup sofia/internal/1001@192.168.1.101 [CS_EXECUTE] [NORMAL_CLEARING]

[DEBUG] switch_channel.c:3034 Send signal sofia/internal/sip:1007@192.168.1.107:5060 [KILL]

[DEBUG] switch_core_session.c:1310 Send signal sofia/internal/sip:1007@192.168.1.107:5060 [BREAK]

[DEBUG] switch_ivr_bridge.c:516 sofia/internal/sip:1007@192.168.1.107:5060 ending bridge by request from read function

[DEBUG] switch_ivr_bridge.c:597 BRIDGE THREAD DONE [sofia/internal/sip:1007@192.168.1.107:5060]

[DEBUG] switch_ivr_bridge.c:622 Send signal sofia/internal/1001@192.168.1.101 [BREAK]

[DEBUG] switch_core_state_machine.c:480 (sofia/internal/sip:1007@192.168.1.107:5060) State EXCHANGE_MEDIA going to sleep

[DEBUG] switch_core_state_machine.c:415 (sofia/internal/sip:1007@192.168.1.107:5060) Running State Change CS_HANGUP

97. FreeSwitch 实网系统或者投产系统有哪些配置需要注意？

实网系统或者投产系统，可能需要支持大用户并发通话，可能需要支持用户长时间通话等等各种超大超长的情况。

FreeSwitch 默认的这些参数通常都不够给力。

1.修改 switch.conf.xml 文件的下面参数：

```
<!-- Maximum number of simultaneous DB handles open -->
<param name="max-db-handles" value="500"/>
<!-- Minimum idle CPU before refusing calls -->
<param name="min-idle-cpu" value="2"/>
<!-- Max number of sessions to allow at any given time. -->
<param name="max-sessions" value="10000"/>
<!--Most channels to create per second -->
<param name="sessions-per-second" value="300"/>
```

2.修改 internal.xml 文件的下面参数：

```
<!--max number of open dialogs in proceeding -->
<param name="max-proceeding" value="10000"/>
<!--session timers for all call to expire after the specified seconds -->
<param name="session-timeout" value="18000"/>
<!-- rtp inactivity timeout -->
<param name="rtp-timeout-sec" value="18000"/>
<param name="rtp-hold-timeout-sec" value="18000"/>
```

上面的修改可以支持通话达到 5 小时,即使坐席上线接通之后 5 小时内不说话也不会掉线。

其它说明：

1.假如使用到 external 场景，那么 external.xml 里面对应的也一样要修改

2.rtp-timeout-sec 是 rtp 的超时时间，默认是 300 秒。假如 FreeSwitch 跟软电话通话，软电话接通之后 300 秒内没都没有说话，有些软电话没有说话就不发 rtp，那么 FreeSwitch 会日志提示 MEDIA_TIMEOUT 然后挂机。这种情况在公网运营的呼叫中心坐席长通话连接的使用模式下经常发生。长通话连接是指：坐席迁入然后上线（软电话跟 FreeSwitch 建立通话）之后等待用户来电弹屏，有用户来电可以直接通话，不需要再呼叫坐席软电话。因此实网系统建议改为 18000，也就是 5 个小时或者更加久，否则会发生坐席上线之后掉线情况。当然假如你是短通话连接：每次用户来电都重新呼叫坐席进行通话，通话之后用户挂机，坐席也下线的情况，就不需要修改这个。

98. FreeSwitch 如何使帐号密码保存在 mysql 数据库？

默认用户的认证密码和语音邮箱登陆密码都异常简单（默认为 1234）。

修改密码也可以修改配置文件，但是终归不方便，用户开户开分机的时候每

次都修改配置文件显然不是一个投产的系统，尤其是存在多个 FreeSwitch 的情况下。运维是一个令人恐怖是事情。大多数通过其它版本保存分机帐号和密码。

具体方法可以参考以下建议：

1、 删除默认的静态 XML 配置，通过 mod_xml_curl 模块使用后台数据库中动态的数据。

2、 手动修改配置文件中的用户名和密码。

3、 通过运行 scripts/perl/randomize-passwords.pl 修改。

4、 通过 lua 脚本实现。

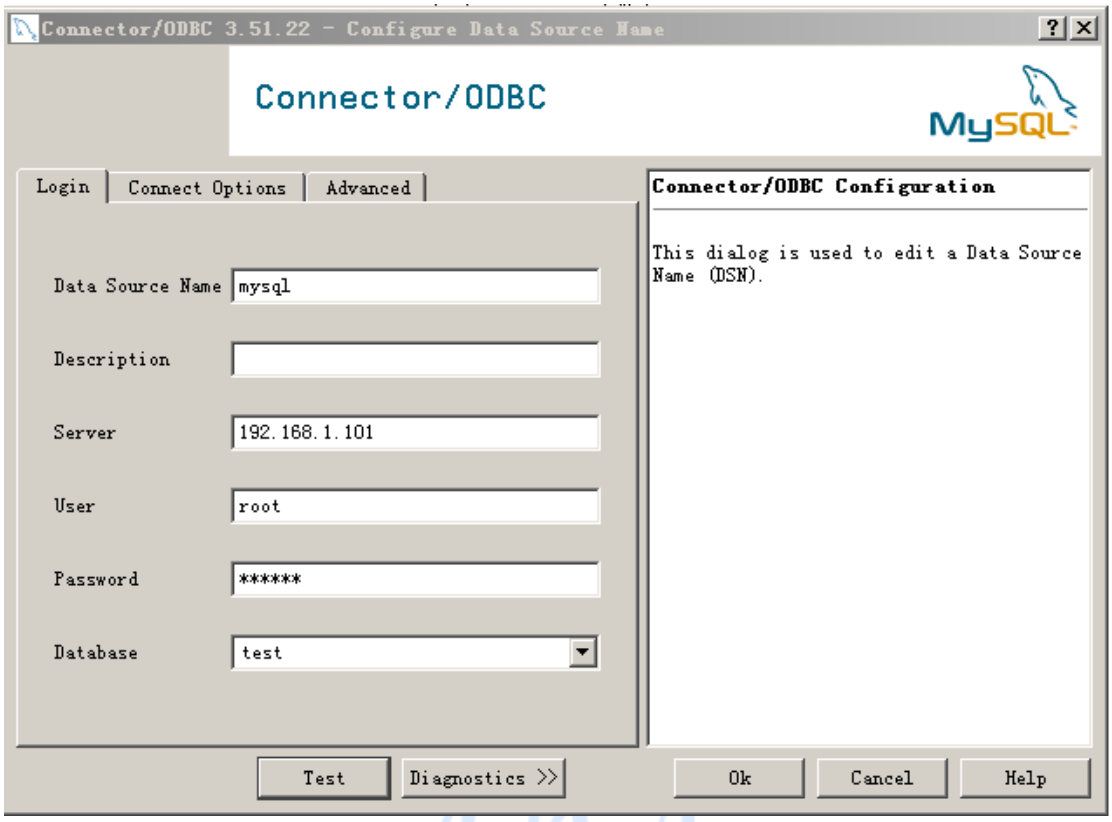
其中通过 lua 脚本实现比较简单，以帐号数据存放在 mysql 数据库为例，其过程说明如下：

1）在 mysql 数据库中建立如下表：

```
CREATE TABLE     userinfo(
  `id` int(11) NOT NULL auto_increment,
   username varchar(64) NOT NULL default '',
   password       varchar(64) NOT NULL default '',
   updatedate       timestamp NOT NULL default CURRENT_TIMESTAMP ,
  PRIMARY KEY    (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=gbk;
```

其中 username 就是分机号码，password 就是分机对应的密码

2）创建 mysql ODBC 数据源，以 windows 版本为例内容如下：



其中的 mysql 是数据源名称，下面要用到。

3）修改 conf\autoload_configs\lua.conf.xml 文件，结果如下：

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
   <param name="xml-handler-script" value="gen_dir_user_xml.lua"/>
   <param name="xml-handler-bindings" value="directory"/>
  </settings>
</configuration>
```

4）创建 gen_dir_user_xml.lua 文件并且存到 \scripts 目录下，gen_dir_user_xml.lua 内容如下：

```
local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
local req_user   = params:getHeader("user")
local dbh = freeswitch.Dbh("mysql","root","123456")
if dbh:connected() == false then
  freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" .. dsn .. "\n")
  return
end
XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type=" freeswitch/xml">
  <section name="directory">
    <domain name="]] .. req_domain .. [[">
      <user id="]] .. req_user  .. [[">
        <params>
```

```
                <param name="password" value=" FreeSwitch+ysyhL9T"/>
                  <param name="dial-string" value="{sip_invite_domain=${dialed_domain},
presence_id=${dialed_user}@${dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
              </params>
              <variables>
                <variable name="user_context" value="default"/>
              </variables>
            </user>
          </domain>
        </section>
      </document>]]
      local my_query = string.format("select password from userinfo where username='%s' limit 1", req_user)
      assert (dbh:query(my_query, function(u) -- there will be only 0 or 1 iteration (limit 1)
        XML_STRING =
      [[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
      <document type=" freeswitch/xml">
        <section name="directory">
          <domain name="]] .. req_domain .. [[">
            <user id="]] .. req_user    .. [[">
              <params>
                <param name="password" value="]] .. u.password .. [["/>
                  <param name="dial-string" value="{sip_invite_domain=${dialed_domain},
presence_id=${dialed_user}@${dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
              </params>
              <variables>
                <variable name="user_context" value="default"/>
              </variables>
            </user>
          </domain>
        </section>
      </document>]]
      end))
```

说明：其中"mysql" 是数据源名称，"root" 是帐号，"123456" 是 mysql 的 root 帐号的密码。其中“FreeSwitch+ysyhL9T”是系统的超级密码，即使数据库里面没有也可以使用帐号使用。

99. FreeSwitch 如何配置帐号密码在 oracle 数据库里面？

上一个问题解决了用户的帐号和密码保存在 mysql 数据库里面的问题。

Lua 脚本通过 odbc 去取进行认证。

但是假如帐号和密码保存在 oracle 数据库里面的，那么 FreeSwitch 由于源码的 bug 就不能使用。这个使用就需要修改源码重新编译。

需要修改两个地方：

1. 修改 switch_odbc.c 把里面原来下面的代码：

```
strcpy((char *) sql, "select 1 ");
```

改为：

```
 strcpy((char *) sql, "select 1 from dual");    //yhy2013-03-07 to support oracle database
```

然后重新编译 FreeSwitch。这样的修改可以同时支持 mysql 和 oacle，

不过这样就不能支持 sqlserver 了。

归纳如下：

“select 1” 的语法是 支持 mysql 和 sqlserver。

“select 1 from dual” 的语法是 支持 mysql 和 oracle。

我猜测这个 sql 语法是数据库心跳的握手语法。如何做到同时支持上面 4 种数据库？

可以考虑使用配置参数。

修改之后重新编译源码，使用新的可执行文件。

2. 上面支持 mysql 面的 lua 的脚本也要修正才可以，要修改为下面的脚本：

```
-- gen_dir_user_xml.lua
-- example script for generating user directory XML
-- comment the following line for production:
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, provided params:\n" .. params:serialize() .. "\n")

local req_domain = params:getHeader("domain")
local req_key       = params:getHeader("key")
local req_user      = params:getHeader("user")
local dbh = freeswitch.Dbh("test","system","test")
if dbh:connected() == false then
   freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" .. dsn .. "\n")
   return
end
-- it's probably wise to sanitize input to avoid SQL injections !
local my_query = string.format("select fsname,fspassword from    fsuserinfo where fsname='%s' ",    req_user)
freeswitch.consoleLog("notice", "sql=" .. my_query .. "\n")
local my_pass =" "
local my_query = "select    password    from userinfo where    username = " .. req_user
-- set variable - or print to console if no session is available
local function sv(key, val)
            print(key .. " : " .. val)
  my_pass=val;
end
assert(dbh:query(my_query, function(row)
      for key, val in pairs(row) do        -- in this example only one row with one column will be returned
         sv(key, val)                          -- so here key = 'user'
      end
end))
 XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
   <section name="directory">
      <domain name="]] .. req_domain .. [[">
```

```
            <user id="]] .. req_user    .. [[">
              <params>
                <param name="password" value="]] .. my_pass .. [["/>
                <param name="dial-string" value="{sip_invite_domain=${dialed_domain},
presence_id=${dialed_user}@${dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
              </params>
              <variables>
                <variable name="user_context" value="default"/>
              <variable name="transfer_fallback_extension" value="operator"/>
                <variable name="toll_allow" value="domestic,international,local"/>
              <param name="accountcode" value="]] .. req_user .. [["/>
                <param name="effective_caller_id_number" value="]] .. req_user .. [["/>
                 <param name="effective_caller_id_name" value="]] .. req_user .. [["/>
                </variables>
            </user>
          </domain>
        </section>
      </document>]]
        -- comment the following line for production:
        freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, generated XML:\n" .. XML_STRING .. "\n")
```

注意上面的斜体部分。

100. FreeSwitch 的 SDP 有啥特别和缺陷？

FS 默认的配置下 G711 的 Alaw 和 ulaw 没有 m=rtpmap 0/8000 之类 的详细描述
导致呼叫到老版本的 kapanga 上，kapanga 接通就挂机。
虽然呼叫到新版本的 kapanga（ver 1.00.2180b 之后的版本）上是可以的.
另外，没有 rtpmat 描述 导致呼叫到 red5 phone 等不行
这样也说明一个问题：

FreeSwitch 的 sip sdp 的兼容性还有需要改进的地方。否则跟有些做得不是特别完善的 sip 软电话的兼容性匹配就无法做到 100%兼容。这缺陷通过在 vars.xml 里面加上 verbose_sdp=true 可以解决上面的问题。

另外一个特别的地方是：跟其它的软电话之类的对比 多了 rtpmap=13 静音的说明。

101. 两个 FreeSwitch 如何做集群？

多个 FreeSwitch 服务器如何做集群有很多方法，下面这个是基于 ACL 的方法：
假设你有 FreeSwitch A (192.168.1.130) 分机 1000-1099
和 FreeSwitchB (192.168.1.236) 分机 1100-1199.
首先我们要修改 autoload_configs/acl.conf.xml
FreeSwitchA 上在 <list name="domains" ...>上增加：

<node type="allow" cidr="192.168.1.236/32"/>

FreeSwitchB 上在 <list name="domains" ...>上增加：

```
<node type="allow" cidr="192.168.1.130/32"/>
```

然后在 FreeSwitchA 上的 conf/dialplan/default.xml 里面增加：

```
<extension name="Dial to FreeSwitchB">
  <condition field="destination_number" expression="^(11\d+)$">
    <action application="bridge" data="sofia/internal/$1@192.168.1.236"/>
  </condition>
</extension>
```

表示 11XX 的分机会到 FreeSwitchB 的 IP 上

在 FreeSwitch B 上的 conf/dialplan/default.xml 里面增加：

```
<extension name="Dial to FreeSwitchA">
  <condition field="destination_number" expression="^(10\d+)$">
    <action application="bridge" data="sofia/internal/$1@192.168.1.130"/>
  </condition>
</extension>
```

表示 10XX 的分机会到 FreeSwitch A 的 IP 上

最后

在 FreeSwitch A 的 conf/dialplan/public.xml 里面增加

```
<extension name="Calls from FreeSwitchB">
  <condition field="destination_number" expression="^(10\d+)$">
    <action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

这个表示把 FreeSwitch B 打过来的来电分机号码转到 XML default 从而进入正常的拨号计划

在 FreeSwitch B 的 conf/dialplan/public.xml 里面增加

```
<extension name="Calls from FreeSwitchA">
  <condition field="destination_number" expression="^(11\d+)$">
    <action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

这个表示把 FreeSwitch A 打过来的来电分机号码转到 XML default 从而进入正常的拨号计划,正常的拨号计划如下（记得要放在 Dial to FreeSwitchB 或者 Dial to FreeSwitchA 的拨号计划之后）：

```
<extension name="Local_Extension2">
  <condition field="destination_number" expression="^([0-9]\d+)$">
  <action application="export" data="dialed_extension=$1"/>
  <action application="set" data="call_timeout=30"/>
  <action application="set" data="record_sample_rate=8000"/>
  <action application="export" data="RECORD_STEREO=false"/>
  <action application="set" data="hangup_after_bridge=true"/>
  <action application="set" data="continue_on_fail=true"/>
  <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
  </condition>
</extension>
```

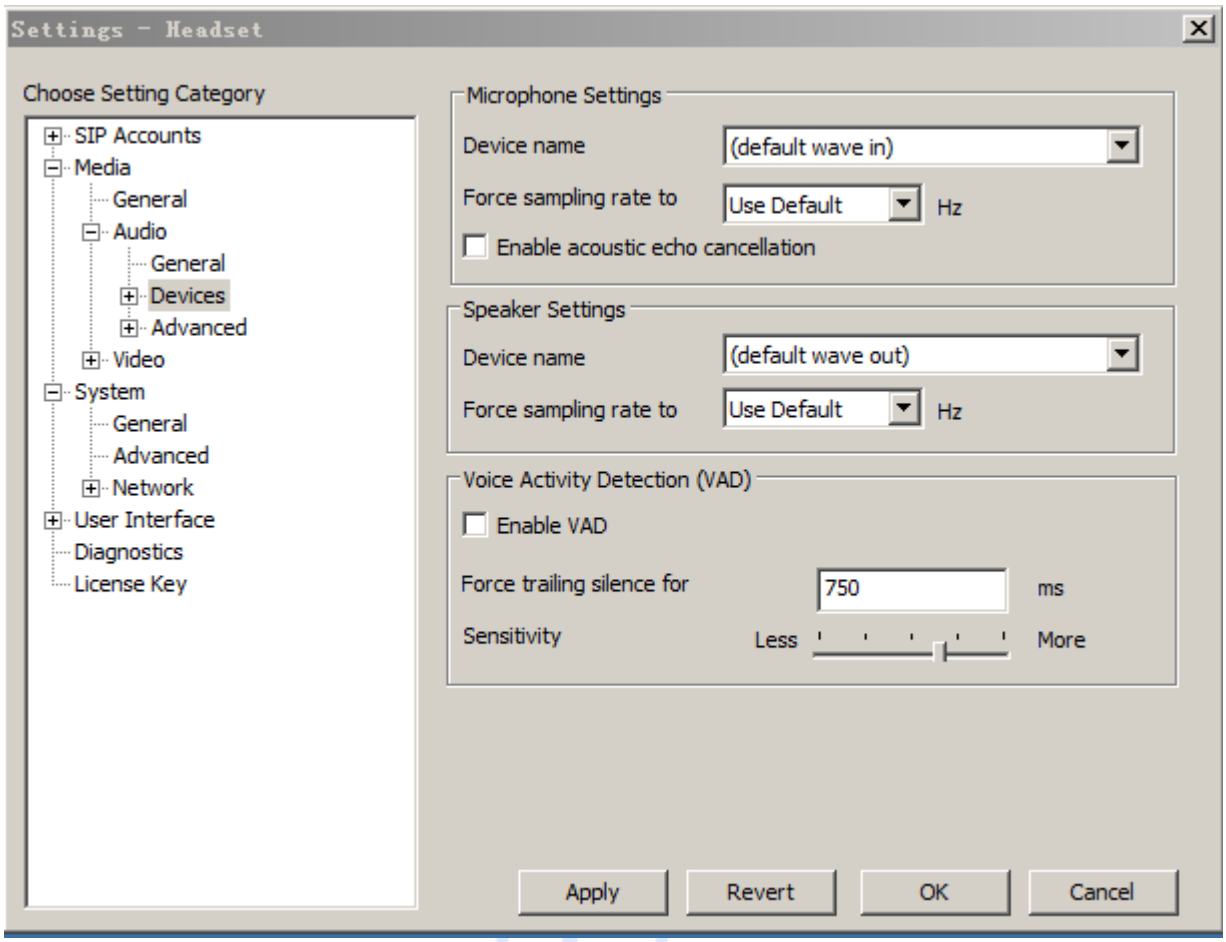
102. FreeSwitch 如何使 VAD 真正生效，以节省带宽？

首先设置 设置参数 internal.xml 里面的 vad 参数：

<param name="vad" value="out"/>

或者 <param name="vad" value="both"/>

测试的时候需要先禁用软电话的 VAD，这样不说话软电话才会发静音数据包给 FreeSwitch，以 eyebeam 为例:



里面的 Enable VAD 的打勾需要去掉，默认是打勾的。

测试双方都没有说话测试还有 RTP 包发到客户端，观察 alaw 的 rtp 包都想 0x55，0xd5 之类的静音数据， 浪费了网络带宽

经过代码跟踪发现可能是一个 BUG，修改如下：

switch_rtp.c 里面的

```
static int rtp_common_write(switch_rtp_t *rtp_session,
                            rtp_msg_t *send_msg, void *data, uint32_t datalen, switch_payload_t payload,
uint32_t timestamp, switch_frame_flag_t *flags)
{
```

函数里面：将

```
    if (switch_test_flag(rtp_session, SWITCH_RTP_FLAG_VAD) && rtp_session->recv_msg.header.pt ==
    rtp_session->vad_data.read_codec->implementation->ianacode ) {
```

改为：

//yhy2013-10-30 alaw 测试 vad=both 512,recv_msg.header.pt=13 CN!=iana=8 alaw? 导致 VAD 不起作用
if (switch_test_flag(rtp_session, SWITCH_RTP_FLAG_VAD) //&&
//rtp_session->recv_msg.header.pt == rtp_session->vad_data.read_codec->implementation->ianacode
){
运行修改过的测试 FreeSwitch，测试双方没有说话，FreeSwitch 就不会发 RTP 包给对方。

103. FreeSwitch 如何同时支持 SIP INFO 和 RFC2833 取按键？

Fs 默认的配置是支持 RFC2833 取按键，但是有时候 fs 需要使用 SIP info 方式跟别人对接取按键
这个时候伙计们需要设置
修改 internal.xml 里面的下面参数：
<action application="set" data="dtmf_type=info"/>
在某些场景下，比如有一个落地使用 sip info，另外一个落地使用 rfc2833，这个时候 fs 需要同时支持这两种取按键的方式，这个时候需要
修改 internal.xml 里面的下面参数：
<action application="set" data="dtmf_type=info"/>
<param name="liberal-dtmf" value="true"/>
这样 DTMF 就可以同时支持 sip info 和 RFC2833 的方式获取。

104. FreeSwitch 如何支持单一注册？

单一注册：
<param name="max-registrations-per-extension" value="1"/> 第 2 个注册就会提示 Forbidden
但是无法呼叫
要达到能呼叫
有两种办法：
1. <list name="domains" default="deny"> 改为
<list name="domains" default="allow"> 放开所有的认证，不注册都能拨打。风险很大

2. 比较安全：
<list name="domains" default="deny">
<!-- domain= is special it scans the domain from the directory to build the ACL -->
<node type="allow" domain="$${domain}"/>
<!-- use cidr= if you wish to allow ip ranges to this domains acl. -->
<node type="allow" host="192.168.1.0" mask="255.255.255.0"/>
</list>
只是放开一个内网的网段。
然后主叫通过 p2p 到 fs 再到分机上
3.呼叫通过 192.168.1.100:5080 到达分机
但是这样分机拨打分机不行
结论：建议采用第二种方案

105. FreeSwitch 如何支持单一呼叫？

作为主叫只能拨打一个被叫：修改下面参数：

vars.xml

```
<X-PRE-PROCESS cmd="set" data="max_calls=1"/>
<extension name="limit_exceeded">
    <condition field="destination_number" expression="^limit_exceeded$">
      <action application="answer"/>
      <action application="playback" data="d:/data/system/busy.alaw"/>
      <action application="hangup"/>
    </condition>
</extension>
 <extension name="limit" continue="true">
  <condition>
    <action application="limit" data="db $${domain} ${sip_auth_username} ${max_calls}"/>
  </condition>
</extension>
```

或者：

```
<extension name="limit" continue="true">
    <condition>
      <action application="limit" data="hash $${domain} ${sip_auth_username} ${max_calls}"/>
    </condition>
</extension>
```

作为被叫，只能被单一呼叫：

```
 <extension name="limit" continue="true">
  <condition>
  <action application="limit" data="hash $${domain} ${destination_number} ${max_calls}"/>
  </condition>
</extension>
```

同时作为主叫和被叫单一呼叫：

```
 <extension name="limit" continue="true">
  <condition>
<action application="limit" data="hash $${domain} ${sip_auth_username} ${max_calls}"/>
  </condition>
</extension>
 <extension name="limit" continue="true">
  <condition>
  <action application="limit" data="hash $${domain} ${destination_number} ${max_calls}"/>
  </condition>
```

```
</extension>
```

106. FreeSwitch 如何拨出如何指定参数？

拨出时候指定主叫：

```
<action application="bridge" data="{origination_caller_id_number=1234}sofia/gateway/gw1/$1"/>
```

拨出指定编码器：

```
<action application="bridge" data="{absolute_codec_string='G729'}sofia/gateway/gw1/$1"/>
```

指定显示的 user agent：

```
<action application="bridge" data="{ sip_contact_user=test}sofia/gateway/gw1/$1"/>
```

107. FreeSwitch 如何直接呼叫某个 ip？

直接通过 ip 地址 p2p，拨打外拨网关 不需要 gw1.xml 配置文件

方法 1.随便写一个 gw1.xml:

```
<gateway name="gw1">
    <param name="realm" value="192.168.1.198"/>
    <param name="username" value="5678"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
</gateway>
```

外拨的时候带上地址： 1008@192.168.1.199 根据代码 以 1008@192.168.1.199 里面的 Ip 为主

```
<action application="bridge" data="sofia/gateway/gw1/1008@192.168.1.199"/>
```

方法 2.

被叫号码后面直接@ip，然后使用 external profile

```
<action application="bridge" data="sofia/external/1009@192.168.1.199"/>
```

108. FreeSwitch 如何设置媒体绕过模式？

媒体绕过也叫做 bypass mode

需要设置下面的参数：

```
<param name="inbound-bypass-media" value="true"/>
```

注意：

假如拨号计划里面先应答

有<action application="answer" /> bypass 就不行，变成了媒体代理。

109. FreeSwitch 如何设置视频会议里的主席？

Fs 的视频会议默认只能看某个人，基本上是谁说话的声音大就切换看谁。
这个在实际使用中是不合适的。
以 1.4.X 的版本为例，需要下面两个步骤来达到使用：

1. 去掉谁说话的声音大就切换看谁的功能
   修改下面的代码
   mod_conference.c 里面

```
 if (switch_test_flag(imember, MFLAG_RUNNING) && imember->session) {
            switch_channel_t *channel = switch_core_session_get_channel(imember->session);
      //yhy2018-08-12 not auto floor_holder,we change floor_holder manual
   if (!floor_holder)
   {
       floor_holder = imember;
   switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_WARNING, "yhy set floor_holder to %d\n",
   floor_holder);
   }
```

2. 手工设置可以固定看某个人
   这个有两种办法，

A. 修改 conference.conf.xml，用户可以自己按键设置，比如按#设置自己是主持人或者主席：

```
  <caller-controls>
    <group name="default">
      <control action="vid-floor-force" digits="#"/>
    </group>
  </caller-controls>
```

这个缺点是谁都可以自己设置自己是主席，不符合常理。
B. 在 esl 里面通过代码设置：

```
sprintf(cmd_tmp,"%d vid-floor %d force",userevent.data,member_id);
sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
esl_send_recv(handle, tmp);
```

然后流程控制 这个 是只有主持人才能执行的，控制谁是主席，比较符合常理，但是麻烦点。

110. FreeSwitch 如何处理 bridge 呼叫对方不成功的情况？

在一个用户跟另外一个用户 bridge 之后，假如对方不通，
根据可以 根据 ${originate_disposition} 对呼叫结果进行处理:
具体原因可以看：
http://wiki.freeswitch.org/wiki/Hangup_causes
比如

USER_BUSY 用户忙

CALL_REJECTED 用户拒接
USER_NOT_REGISTERED 用户没有注册
但是不一定准确，比如
eyebeam 拒接：是返回 NO_USER_RESPONSE ，
超时没有接：是返回 NO_ANSWER。
比如构造下面的拨号计划：

```
<extension name="Local_Extension2">
        <condition field="destination_number" expression="^([0-9]\d+)$">
<action application="export" data="dialed_extension=$1"/>
<action application="set" data="call_timeout=30"/>
<action application="set" data="continue_on_fail=true"/>
 <action application="export" data="record_sample_rate=8000"/>
 <action application="export" data="RECORD_STEREO=false"/>
 <action application="export" data="hangup_after_bridge=true"/>
<action application="bridge" data="user/$1@${domain_name}"/>
<action application="transfer" data="play-cause-${originate_disposition}"/>
</condition>
    </extension>

 <extension name="Local_Extension_error1">
        <condition field="destination_number" expression="^play-cause-USER_NOT_REGISTERED$">
        <action application="answer" />
 <action application="playback" data="d:/data/system/ext_err1.alaw"/>
<action application="hangup" />
 </condition>
    </extension>

<extension name="Local_Extension2">
        <condition field="destination_number" expression="^play-cause-NO_ANSWER$">
        <action application="answer" />
 <action application="playback" data="d:/data/system/ext_err2.alaw"/>
<action application="hangup" />
 </condition>
    </extension>

    <extension name="Local_Extension_error3">
        <condition field="destination_number" expression="^play-cause-USER_BUSY$">
        <action application="answer" />
 <action application="playback" data="d:/data/system/ext_err3.alaw"/>
<action application="hangup" />
 </condition>
    </extension>
```

```
  <extension name="Local_Extension_error4">
        <condition field="destination_number" expression="^play-cause-CALL_REJECTED$">
        <action application="answer" />
 <action application="playback" data="d:/data/system/ext_err4.alaw"/>
<action application="hangup" />
 </condition>
     </extension>


<extension name="Local_Extension_error0">
        <condition field="destination_number" expression="^play-cause-(.*)$">
        <action application="answer" />

 <action application="playback" data="d:/data/system/ext_err0.alaw"/>
<action application="hangup" />
 </condition>
     </extension>
```

这样就可根据被叫不同的情况，播放不同的提示语音。

111. FreeSwitch 该选择何种方式落地和选择哪种落地设备？

这个问题比较大比较复杂具体细分，考虑落地有几个因素需要考虑：

1. 是选择数字 E1 还是模拟电话线路
   这个不是技术问题，是政策或者老板钱的问题。
   模拟线路一般情况下做拨入是可以的，做拨出需要开通反极性检测才能知道拨出的时候用户是否接通。
   因此是不建议使用模拟线路，除非一开始的系统功能测试。
   实际投产使用建议开头 E1 线路，选择 7 号信令，或者 ISDN PRI 信令。
2. 是选择 E1 还是 IMS 线路做落地
   有 IMS 当然选择 IMS，直接对接 FS，成本是最低的。
   这个是毫无疑问的。问题是现在许多地方运营商还没有放开 IMS。
   或者放开的运营商费用比较高，而使用 E1 的运营商通信费用比较低，这个就要自己权衡。
3. 是自己做落地出局，还是使用别人的落地？
   自己做落地需要拉 E1，假如量很大，跟运营商费用可以谈的很低，可以自己做落地。好处是稳定性比较高。缺点是投资费用高。
   使用别人的落地好处是便宜，投资少，缺点是别人系统的可靠性你是未知的，可靠性可能是 5 个 9 也可能只有 3 个 9.另外通过公网，网络稳定性和话音质量都要打折扣。

4. 是否要透传任意主叫号码？
   这个跟业务相关，假如是要拨出的时候透传任意主叫，这个现在运营商基本是不开放的。只能考虑通过别人的落地来透传。
   现在运营商能做到的基本上是叫做呼叫前转的方式透传。

已经转出的时候设置重定向号码，源被叫之类的。
这样通过 7 号或者 isdn pri 信令都可以支持。但是信令支持不等于所有设备都支持
假如有呼叫前转透传任意主叫的业务，采购设备的时候要跟设备厂家问清楚，是否支持呼叫前转的方式透传任意主叫，我曾经使用毅航 ISX 是实现 7 号和 ISDN PRI 的呼叫前转的方式透传任意主叫。网关设备鼎信通达据说也支持，但是我未测试确认。

5. 是否要允许用户回电
假如要允许来电，就不能透传随便的号码，否则可以透传任意号码，透传任意主叫号码意味着下一个来电的问题，现在一般的做法是透传 400 号码，然后 400 来电绑定到系统的中继号码上。

6. 是否使用 SIP TRUNK 方式对接
一般自己做的落地都是使用 SIP TRUNK 进行对接
Fs 这边配置网关的 gw。使用不注册的方式
比如 3 个网关，配置文件如下：
conf\sip_profiles\external\gw1.xml

```
<gateway name="gw1">
    <param name="realm" value="10.10.10.47:5060"/>
    <param name="username" value="1000"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
    <param name="caller-id-in-from" value="true"/>
</gateway>
<gateway name="gw2">
    <param name="realm" value="10.10.10.48:5060"/>
    <param name="username" value="1000"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
    <param name="caller-id-in-from" value="true"/>
</gateway>
<gateway name="gw3">
    <param name="realm" value="10.10.10.49:5060"/>
    <param name="username" value="1002"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
    <param name="caller-id-in-from" value="true"/>
</gateway>
```

然后网关那边配置 fs 这边的 ip 和端口，由于使用 SIP TRUNK 方式对接直接配置到 5080 端口就可以使用。假如不是 SIP TRUNK 方式对接，就要配置到 fs 的 5060 端口，然后要使用分机的注册方式进行注册。

7. 对接使用啥编码器
一般内网对接可以直接使用 G711 的 alaw 语音编码，但是注意假如量大的话要上千兆口到 FS 的服务器，目前网关的端口大多还是百兆端口。一般一个网关 240 线 100Mb 的带宽跑 G711 肯定够用。但是 FS 一般可以支持很多台网关，这样的话 100Mb 就可能不够用，因此需要千兆端口。服务器现

在都是千兆端口。这个网卡现在不是问题。

8. 对接使用啥设备

落地的设备是跟上面的问题相关的，使用 IMS 或者别人的落地就无需落地设备。

假如使用 E1 落地，

可以选择的设备厂家有：

毅航 ISX1000

迅时 E1 网关

鼎信通达 E1 网关

网经 E1 网关

Sangoma 板卡。

从成本和容量上看，Sangoma 板卡是最划算的。本书的第十二章专门有一个章节介绍这个板卡。

板卡便宜，缺点是不灵活要插在机器上。

这里主要讨论网关：

综合看来

鼎信通达 E1 网关

网经 E1 网关

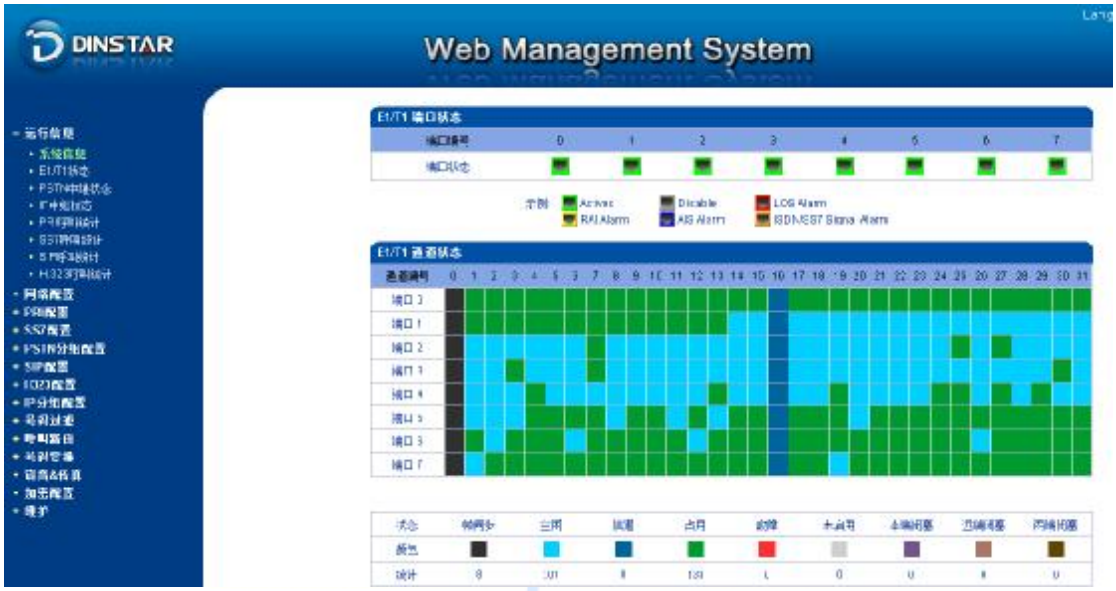
性价比比较高

鼎信通达 E1 网关和网经 E1 网关都可以单机到 8E1

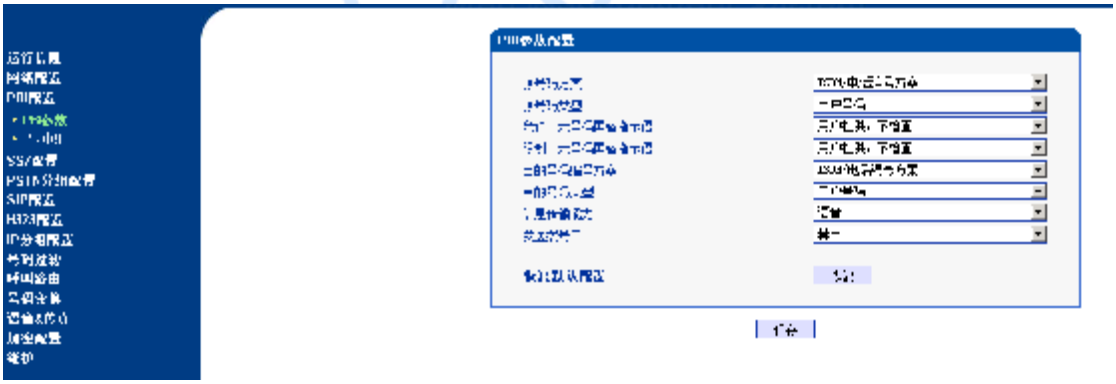
鼎信通达网关内部结构如下：

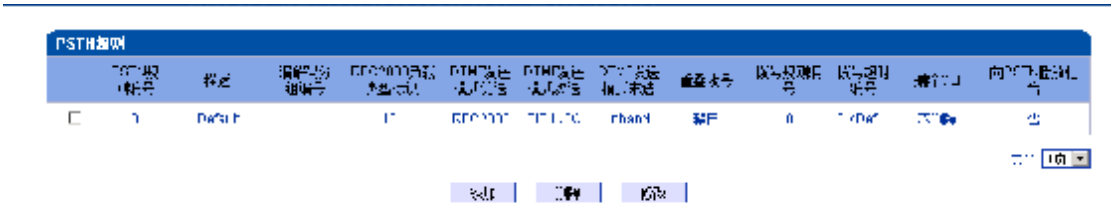
看到了热备的双电源，该给一个点赞

下面这个是鼎信通达用于一个投产系统的监控界面。
可以看到实时并发的情况，运营管理比较方便：



以 ISDN 为例
PSTN 的配置主要就下面几个页面：





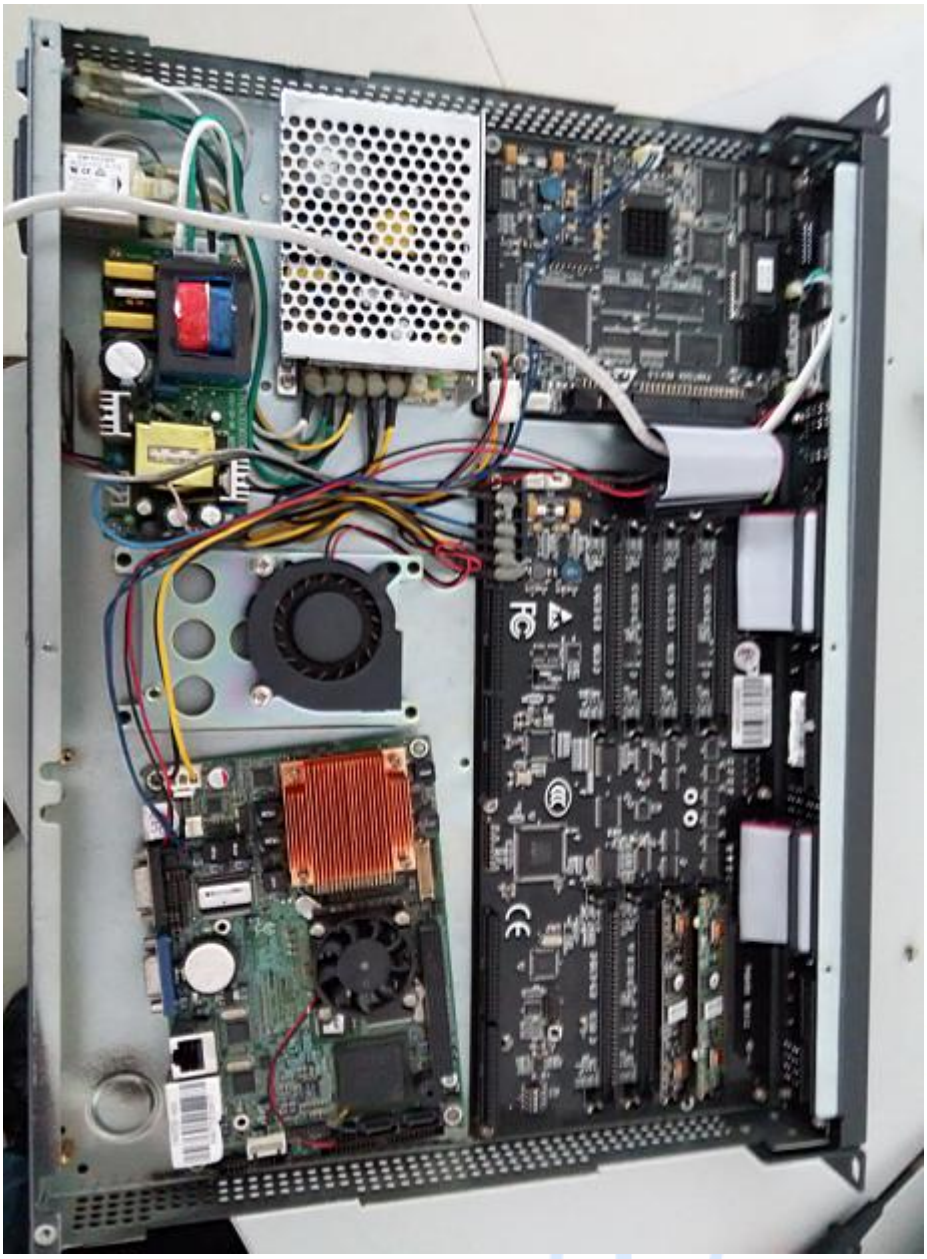
IP 的配置主要是下面这几个页面

然后就是路由配置：

网经 E1 网关内部结构如下：

只有一个电源，作为高可靠性设备，一个电源是不靠谱的，应该改进，我已经给厂家建议了。
具体的配置可以参考他们的配置手册。

关于设备电源我再扯淡几句：
我碰到实网投产系统使用某进的一体机设备连续几台都是因为电源问题导致机器设备无法启动，只能派人去现场更换设备。下面这个是拆回来的设备我拆开折腾的照片，里面是单电源供电，研究了很久才知道是电源供电不足导致的问题，吐槽一下：



另外假如你选择 Sangoma 板卡作为落地的时候，建议大家选择 Pc server 采购的时候也必须是冗余电源和两个硬盘做 RAID1。

第五章 FreeSwitch 性能测试部分

112. FreeSwitch 对内存 cpu 需求如何？

基本的机器就可以，目前主流的双 四核的 cpu，8G 内存配置的 pc Server 机器，可以支持 300 先线并发带转码。
在 B950 CPU 2G 内存配置的笔记本上测试可以支持 30 路的转码（G.711--SPEEX）。
假如内存不够，会导致 FreeSwitch 崩溃。
在 3612 QM CPU4G 内存配置的笔记本上测试可以支持 200 路的 ESL IVR。

113. FreeSwitch 32 位版 windows 下单机最大支持多少线并发通话？

FreeSwitch 的源码是支持 32 位和 64 位的，但是发布包里面的 windows 下工程只有 32 位的工程。
因此首先测试 32 位编译的版本。编译环境是 VS2010 旗舰版。
测试服务器使用 dell R720 型号的机器，
机器配置：E5-2640cpu*2， 16G 内存，500G 硬盘*2 做 raid1，价格大约 2.7 万
操作系统是 windows2008 64 位 R2 的版本。
测试的 FreeSwitch 版本是 1.2.10 release 编译的版本。
测试场景是分机到分机的通话，媒体经过 FreeSwitch 转发，语音统一使用 ulaw 编码，没有使用视频，FreeSwitch 没有做媒体编码转换，只有转发。
测试客户端是自己编写的 pjsip 客户端。
switch.conf.xml 配置

```
<param name="max-db-handles" value="500"/>
<param name="db-handle-timeout" value="10"/>
<param name="min-idle-cpu" value="5"/>
<param name="max-sessions" value="10000"/>
<param name="sessions-per-second" value="300"/>
```

测试到 900 线并发的时候，Cpu 大约 25%。
假如是使用 VS2010 32 位编译（X86 模式，发布包里面 vs2010 的工程文件默认的模式就是 X86）编译的话，再继续测试增加压力，接近 1000 线时候，就开始不稳定了，
会出现：

```
[CRIT] switch_core_session.c:1697 Thread Failure!
switch_core_session.c:1657 LUKE: I'm hit, but not bad.
LUKE'S VOICE: Artoo, see what you can do with it. Hang on back there....
```

这个时候 FreeSwitch 会自动减小 max-sessions 的数目（太变态了，也就是说你配置的 MAX-SESSION 参数被强制动态修改了）。
导致后续的压力测试 出现：

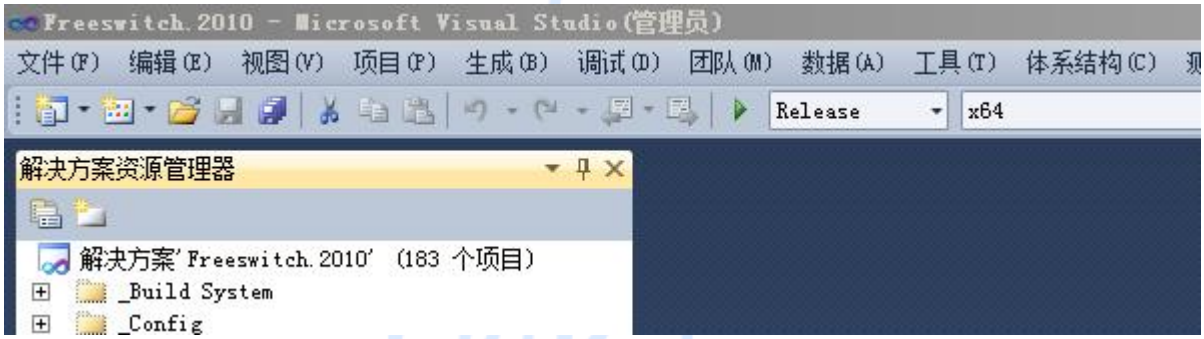
```
Over Session Limit!
```

错误

添加日志发现是 创建线程的时候不成功导致，*GetLastErrorNo* 返回 8 ，没有足够的内存。
结论：编译 X86 的 32 位 FreeSwitch 版本，最大支持稳定的线路并发是 900 线并发。

114. FreeSwitch 64 位版 windows 下单机最大支持多少线并发通话？

要测试 64 位版本，先要有 windows 64 位操作系统的服务器。
测试环境同上面，服务器使用 dell R720 型号的机器，
机器配置 E5-2640cpu*2， 16G 内存，500G 硬盘*2 做 raid1，价格大约 2.7 万
操作系统是 windows2008 64 位 R2 的版本。
测试的 FreeSwitch 版本是 1.2.10 release 编译的版本。
由于 1.2.10 FreeSwitch 默认工程里面没有 64 位的版本，直接把代码的生成 X86 修改为 X64 的不行的。
新的 1.2.23 工程文件里面就有 X64 的版本，直接参照选择下面的 X64 和 Release 就可以编译使用。



而 FS 1.2.10 需要手工建立 64 位的工程：
根据 http://msdn.microsoft.com/en-us/library/9yb4317s
How to: Configure Visual C++ Projects to Target 64-Bit Platforms 描述：

To set up C++ applications to target 64-bit platforms

1. Open the C++ project that you want to configure.
2. Open the property pages for that project. For more information, see How to: Open Project Property Pages.

   Note

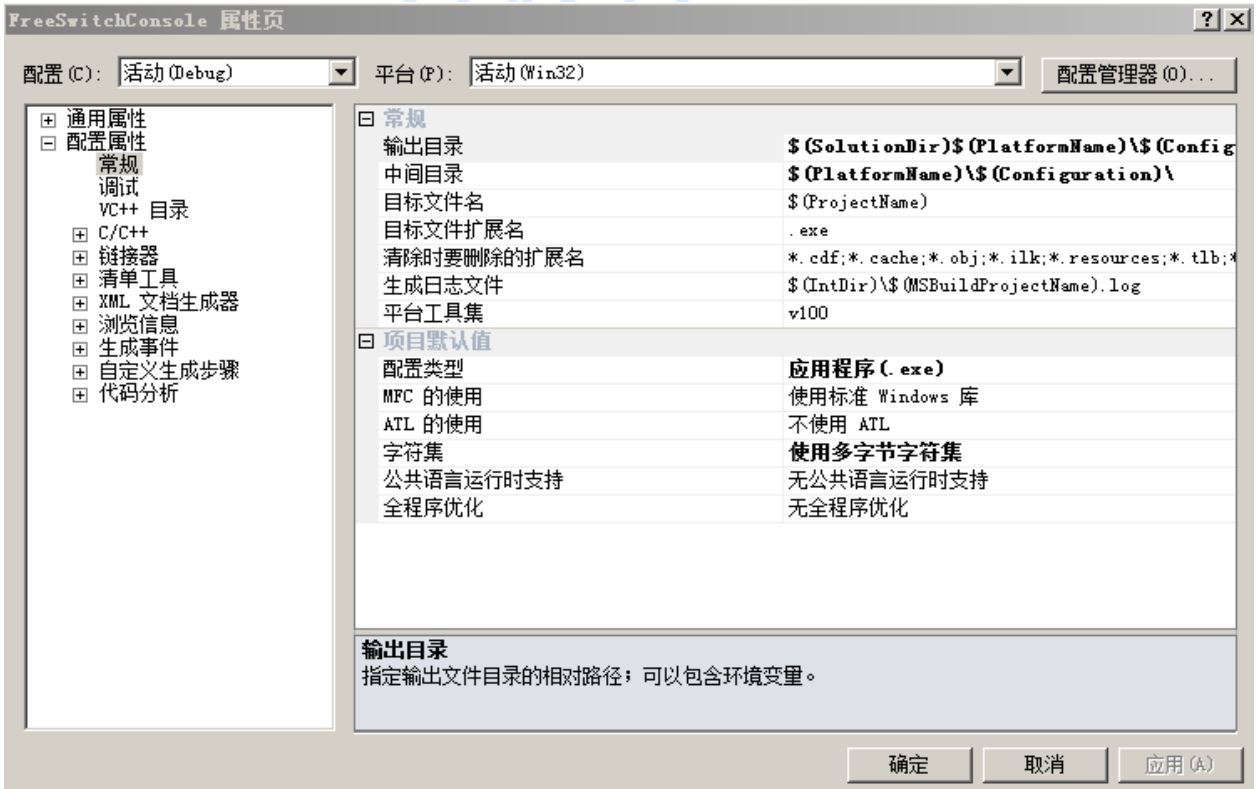
   For .NET projects, make sure that the **Configuration Properties** node, or one of its child nodes, is selected in the **<Projectname> Property Pages** dialog box; otherwise, the **Configuration Manager** button remains unavailable.
3. Choose the **Configuration Manager** button to open the **Configuration Manager** dialog box.
4. In the **Active Solution Platform** drop-down list, select the **<New...>** option to open the **New Solution Platform** dialog box.
5. In the **Type or select the new platform** drop-down list, select a 64-bit platform.

   Note

   In the **New Solution Platform** dialog box, you can use the **Copy settings from** option to copy existing project settings into the new 64-bit project configuration.
6. Choose the **OK** button. The platform that you selected in the preceding step appears under **Active Solution Platform** in the **Configuration Manager** dialog box.
7. Choose the **Close** button in the **Configuration Manager** dialog box, and then choose the **OK** button in the **<Projectname> Property Pages** dialog box.

具体操作如下：

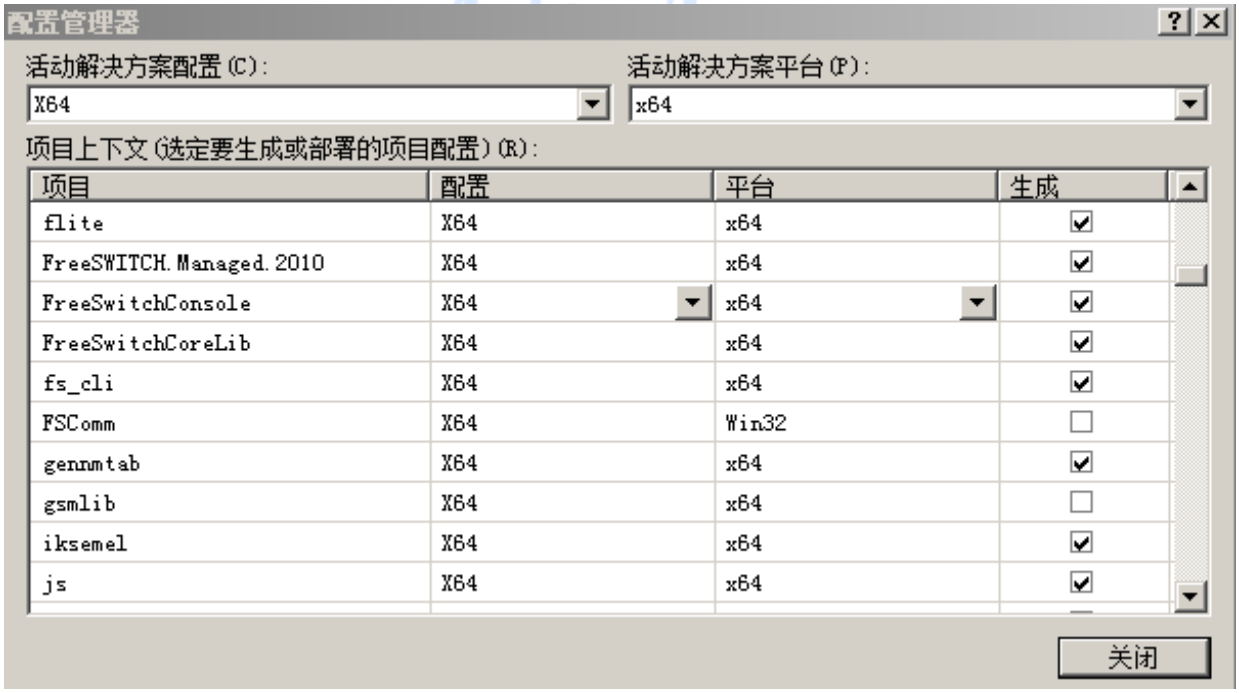
首先 在 FreeSwitchConsole 工程上点右键，然后点属性，如下图：



然后点配置管理器，

然后在【活动解决方案配置】 下拉中点新建，然后输入新建的名称比如 X64，然后在【从此处复制中】选择 Release。如下图：

点确定，最后在【活动解决方案平台】中选择 X64。如下图：



然后点关闭。再点确定。就可以看到下面的两个 X64 的界面。表明已经是 64 位的工程了。

按 F7 重新编译。

产生的 FreeSwitchConsole.exe 等 文件在 FreeSwitch-1.2.10\x64\x64 目录下。

注意，有些文件比如 pthread.dll 和 mod_g729.dll 等产生在 FreeSwitch-1.2.10\x64\Release 目录下，需要自己手工拷贝到 X64 下合并。

另外需要注意的是：

msvcp100.dll 和 msvcr100.dll 文件需要从类似下面的目录

D:\Program Files\Microsoft Visual Studio 10.0\VC\redist\x64\Microsoft.VC100.CRT

中拷贝到 FreeSwitch-1.2.10\x64\x64 目录下

这样 FreeSwitch-1.2.10\x64\x64 目录下 就是完备的，可以发布的版本

这个版本在 windows 2008 64 位 R2 的版本上测试可以超过 1000 线稳定运行，

E5-2640cpu*2，16G 内存配置的机器，通话没有录音的模式下,本人测试最大单机支持超过 2000 线通话稳定运行。

115. 如何使用 sipp 对 FreeSwitch 进行压力测试？

压力测试的工具很多，有 winsip，sipp，也可以自己使用 FreeSwitch 写 esl 发起压力测试呼叫，也可以使用 pjsip 改造成压力测试工具。综合考虑测试工具是 cpu 占用情况，本人最后选定 sipp 作为压力测试工具。

Sipp 的测试工具有 windows 和 linux 下，本人使用 windows 下的版本进行测试。

说明：Sipp windows 版本不支持注册，因此呼叫场景脚本里面不能有注册的场景，假如需要注册的场景，需要使用 linux 下的源码进行编译。幸好 FreeSwitch 支持不注册的主叫呼叫，被叫可以作为网关被呼叫，也不需要注册。Sipp linux 下的安装编译参见：

http://blog.csdn.net/hanruikai/article/details/8024924

a) # tar -xvf sipp-1.1rc6.tar

b) # cd sipp-1.1.rc6

c) # make pcapplay_ossl

测试环境准备：要准备 3 台机器，

假设 sipp 安装在 192.168.1.253 上作为主叫，

假设 sipp 安装在 192.168.1.100 上作为被叫。

假设 FreeSwitch 在 192.168.1.101 上启动,作为被压力测试的东东。

本次测试测试分为两种：

一种是低压模式下的分机对分机的通话压力测试。

一种是高压模式下的分机对分机的通外带全程录音的压力测试。

当然你也可以稍微修改一下就可以做会议模式下的压力测试。

先去下载 sipp-win32-3.1.1.exe。然后安装，

安装之后,假如机器上没有安装过 WinPcap 需要下载安装 WinPcap

：比如 WinPcap_4_0_2.exe 版本。

安装 WinPcap 之后不需要重启可以马上使用。

Sipp 可以压力测试 sip 和媒体 rtp，

为了对 FreeSwitch 进行全面的压力测试。

本人使用了同时压力测试 sip 和媒体 rtp，

具体过程如下：

第一步，先录制 rtp 流的 cap 文件。

启动 FreeSwitch，假设 FreeSwitch 机器的 ip 是 192.168.1.101，不同的机器 a 和 b 上启动两个 eyebeam，呼叫接通之后，eyebeam 设置都只有 ulaw 的语音编码器。

在 b 机器上使用 Ethereal 进行抓包。

条件如下：udp and src host 192.168.1.101

目的是只要抓 192.168.1.101 发出来的包。

然后在 a 机器上的 eyebeam 说话，一直说话，大约持续 150 秒。

然后停止 b 机器的抓包，保存为 1.cap,最后再挂机。

再次在 b 机器上进行通话抓包，这次不是在 a 机器上说话，而是在 a 机器上使用使用 mediaplay 播放音乐，把耳麦开到最大。一直播放，大约持续 150 秒。

然后停止 b 机器的抓包，保存为 2.cap,最后再挂机。

第二步，编写主叫和被叫流程场景 xml 文件。我们的场景设计每通通话持续 150 秒。

主叫 Caller.xml 如下：

```
<?xml version="1.0" encoding="us-ascii"?>
<scenario name="New_Call">
        <send retrans="500">
                <![CDATA[
INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>
Call-ID: [call_id]
CSeq: 1 INVITE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: [len]

v=0
o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
s=-
c=IN IP[media_ip_type] [media_ip]
t=0 0
m=audio [media_port] RTP/AVP 0
a=rtpmap:0 PCMU/8000]]>
        </send>
        <recv response="100" optional="true" />
        <recv response="180" optional="true" />
        <recv response="181" optional="true" />
        <recv response="182" optional="true" />
        <recv response="183" optional="true" />
```

```
        <recv response="200" crlf="true" />
        <send>
            <![CDATA[
ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 1 ACK
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: Performance Test
Content-Length: 0
]]>
        </send>
        <pause milliseconds="1000" />
        <nop>
       <action>
          <exec play_pcap_audio="./1.cap"/>
       </action>
    </nop>
<pause milliseconds="150000" />
        <send retrans="500">
            <![CDATA[
BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
Call-ID: [call_id]
CSeq: 2 BYE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: Performance Test
Content-Length: 0

]]>
        </send>
        <recv response="200" crlf="true" />
        <label id="1" />
        <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200" />
        <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000" />
</scenario>
```

被叫 Called.xml 如下：

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!DOCTYPE scenario SYSTEM "sipp.dtd">
<scenario name="Basic UAS responder">
<recv request="INVITE" crlf="true">
</recv>
<send>
      <![CDATA[
         SIP/2.0 180 Ringing
         [last_Via:]
         [last_From:]
         [last_To:];tag=[call_number]
         [last_Call-ID:]
         [last_CSeq:]
         Contact: <sip:[local_ip]:[local_port];transport=[transport]>
         Content-Length: 0
      ]]>
</send>
<send retrans="500">
      <![CDATA[
         SIP/2.0 200 OK
         [last_Via:]
         [last_From:]
         [last_To:];tag=[call_number]
         [last_Call-ID:]
         [last_CSeq:]
         Contact: <sip:[local_ip]:[local_port];transport=[transport]>
         Content-Type: application/sdp
         Content-Length: [len]

         v=0
         o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
         s=-
         c=IN IP[media_ip_type] [media_ip]
         t=0 0
         m=audio [media_port] RTP/AVP 0
         a=rtpmap:0 PCMU/8000
      ]]>
</send>
<recv request="ACK"    rtd="true" crlf="true">
   </recv>
       <pause milliseconds="1000" />
       <nop>
      <action>
         <exec play_pcap_audio="./2.cap"/>
      </action>
```

```
    </nop>
  <recv request="BYE">
    </recv>
  <send>
      <![CDATA[
        SIP/2.0 200 OK
        [last_Via:]
        [last_From:]
        [last_To:]
        [last_Call-ID:]
        [last_CSeq:]
        Contact: <sip:[local_ip]:[local_port];transport=[transport]>
        Content-Length: 0
      ]]>
  </send>
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
  </scenario>
```

第三步，编写 bat 文件。

Caller1.bat，内容如下：

```
sipp -sn uac 192.168.1.101:5060 -r 4 -rp 1000 -p 15060 -i 192.168.1.253 -s 8253 -sf caller.xml    -l 600
```

参数说明：

192.168.1.101:5060：远端地址和端口,就是要测试的 FreeSwitch 的 sip ip 和 port（在脚本中用[remote_ip]，[remote_port]引入）

-r 4 –rp 1000：每秒钟发 4 个呼，也就是 4caps

-p 15060：本地端口（在脚本中用[local_port]引入）

-i 192.168.1.253：本地地址（在脚本中用[local_ip]引入）

-s 8253：被叫号码（在脚本中用[service]引入）

-sf caller.xml：引入脚本文件，根据需要模拟的呼叫流程编写,通话 150s，然后挂机。

-sn uac :执行默认的 uac 流程,使用脚本这个无效。

-l 600 :最大同时保持呼叫量，假如没有设置，默认值为上面的 4caps 值*脚本里面的 150s 呼叫时长=600，当因种种原因导致现存呼叫总数达到此值时，Sipp 将停止产生新的呼叫，等待现存呼叫总数低于此值时才继续产生呼叫。

Caller2.bat，内容如下：

```
sipp -sn uac 192.168.1.101:5060 -r 4 -rp 1000 -p 25060 -i 192.168.1.253 -s 8254 -sf caller.xml    -l 600
```

说明：写 Caller2.bat:

是为了在 192.168.1.253 上发起两个呼叫场景。从而达到 2*4caps=8caps 的呼叫频率。

Called.bat，内容如下：

```
sipp -sn uac 192.168.1.101:5060 -r 1 -rp 1000 -p 5060 -i 192.168.1.100 -s 2001 -sf called.xml    -l    1200
```

第四步，修改拨号计划和增加网关配置文件:

打开 conf\dialplan\default.xml

把下面这个放在开头的 <include>

<context name="default"> 后面：

```
<extension name="GW_Extension2">
<condition field="destination_number" expression="^(8\d+)$">
    <action application="export" data="dialed_extension=$1"/>
    <action application="set" data="call_timeout=30"/>
    <action application="set" data="record_sample_rate=8000"/>
    <action application="export" data="RECORD_STEREO=false"/>
    <action application="set" data="hangup_after_bridge=true"/>
    <action application="set" data="continue_on_fail=true"/>
    <action application="bridge" data="sofia/gateway/gw1/$1"/>
</condition>
</extension>
```

把下面的内容保存为 gw1.xml，然后放在 conf\sip_profiles\external 目录下

```
<gateway name="gw1">
      <param name="realm" value="192.168.1.100:5060"/>
      <param name="username" value="5678"/>
      <param name="password" value="1234"/>
      <param name="register" value="false" />
</gateway>
```

第五步，验证 rtp 媒体 cap 文件是否正确

在 192.168.1.101 机器上启动 FreeSwitch，

把上面准备好的 1.cap,2.cap.caller.xml,called.xml,caller1.bat,caller2.bat,called.bat 拷贝到 192.168.1.100 和 192.168.1.253 的 sipp 的安装目录下： C:\Program Files\Sipp_3.1

在 192.168.1.100 机器上启动 C:\Program Files\Sipp_3.1 目录下的 startterm.bat

然后在 startterm.bat 环境里面启动 called.bat

然后使用 eyebeam 注册到 FreeSwitch 上，然后拨打 8001，这个时候呼叫会到 100 的机器上，应该能听见 2.cap 里面的声音。临时修改 called.xml 文件把 2.Cap 修改为 1.cap，重启 called.bat，在 eyebeam 上重新拨打 8001 测试 1.cap 里面的声音是否正常。都正常了说明 cap 文件没有问题，可以开始进行压力测试。

第六步，启动压力测试

在 192.168.1.101 机器上启动 FreeSwitch，

把上面准备好的 1.cap,2.cap.caller.xml,called.xml,caller1.bat,caller2.bat,called.bat 拷贝到 192.168.1.100 和 192.168.1.253 的 sipp 的安装目录下： C:\Program Files\Sipp_3.1

在 192.168.1.100 机器上启动 C:\Program Files\Sipp_3.1 目录下的 startterm.bat

然后在 startterm.bat 环境里面启动 called.bat

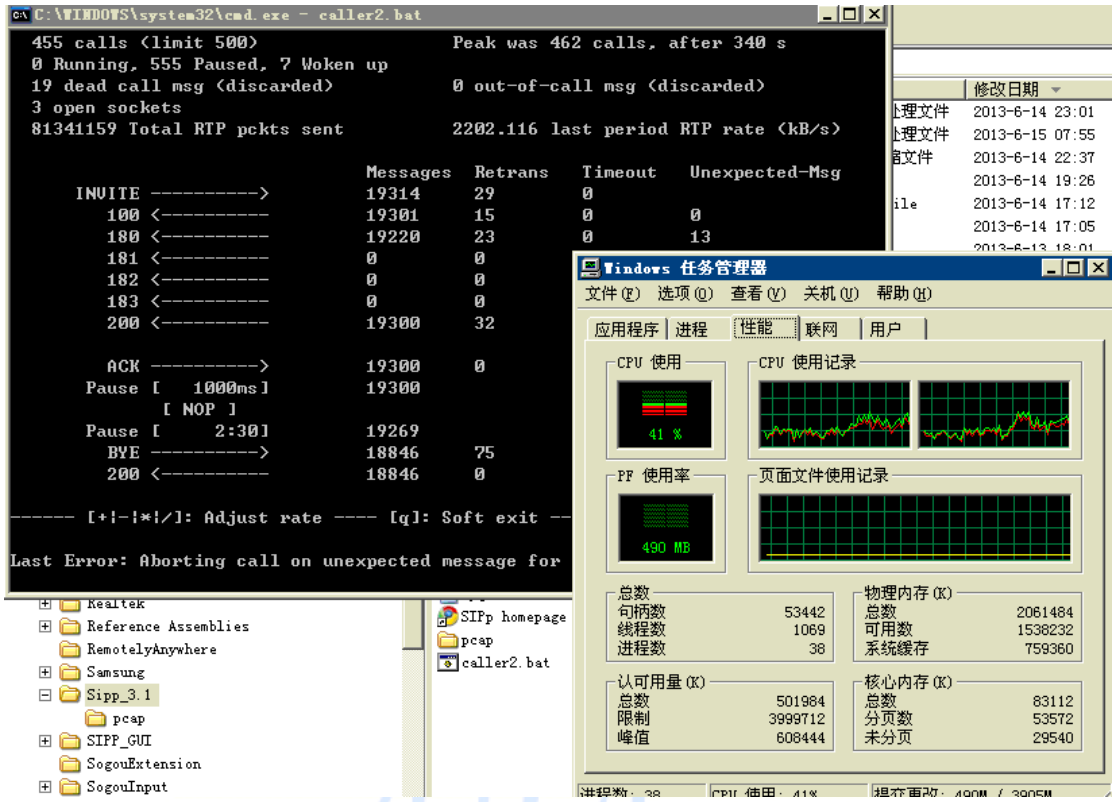
在 192.168.1.253 机器上启动 C:\Program Files\Sipp_3.1 目录下的 startterm.bat

然后在 startterm.bat 环境里面启动 caller1.bat

再次在 192.168.1.253 机器上启动 C:\Program Files\Sipp_3.1 目录下的 startterm.bat

然后在 startterm.bat 环境里面启动 caller2.bat

Sipp 的界面如下：里面有显示 sip 的呼叫过程。已经媒体 rtp 的流量。



压力测试，FreeSwitch 运行在 E5-2640cpu*2，16G 内存配置的机器，通话没有录音的模式下：
本人测试最大单机支持超过 2000 线通话，达到 2488 通话，
偶尔会出现下面错误：

```
2013-06-15 08:11:52.786089 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-15 08:11:52.786089 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-15 08:11:52.786089 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-15 08:11:52.786089 [CRIT] mod_sofia.c:4831 Error Creating Session
```

或者出现这个错误：

```
2013-06-15 08:45:10.746089 [ERR] sofia_presence.c:1667 Presence queue overloaded.... Flushing queue
2013-06-15 08:45:10.766089 [ERR] sofia_presence.c:1667 Presence queue overloaded.... Flushing queue
```

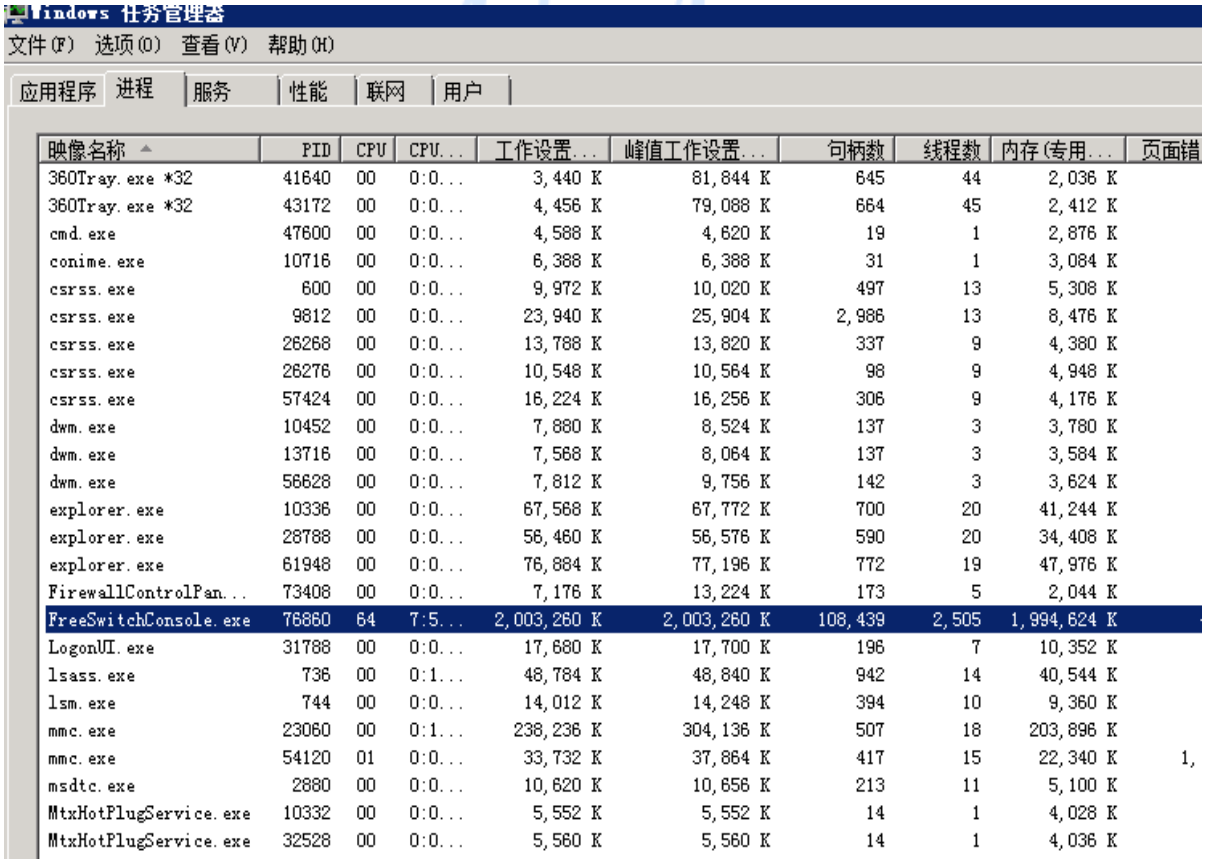
此时整个系统包括 os 负载都很大，任何的 FreeSwitch 命令操作可能都会导致错误输出，比如 show channels 有可能导致下面的错误输出：

```
2013-06-15 08:36:42.746089 [ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [database is locked]
BEGIN EXCLUSIVE
2013-06-15 08:36:42.746089 [CRIT] switch_core_sqldb.c:1712 ERROR [database is locked]
2013-06-15 08:36:42.746089 [ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [cannot commit - no transaction is active]
COMMIT
```

show channels 结果如下图：

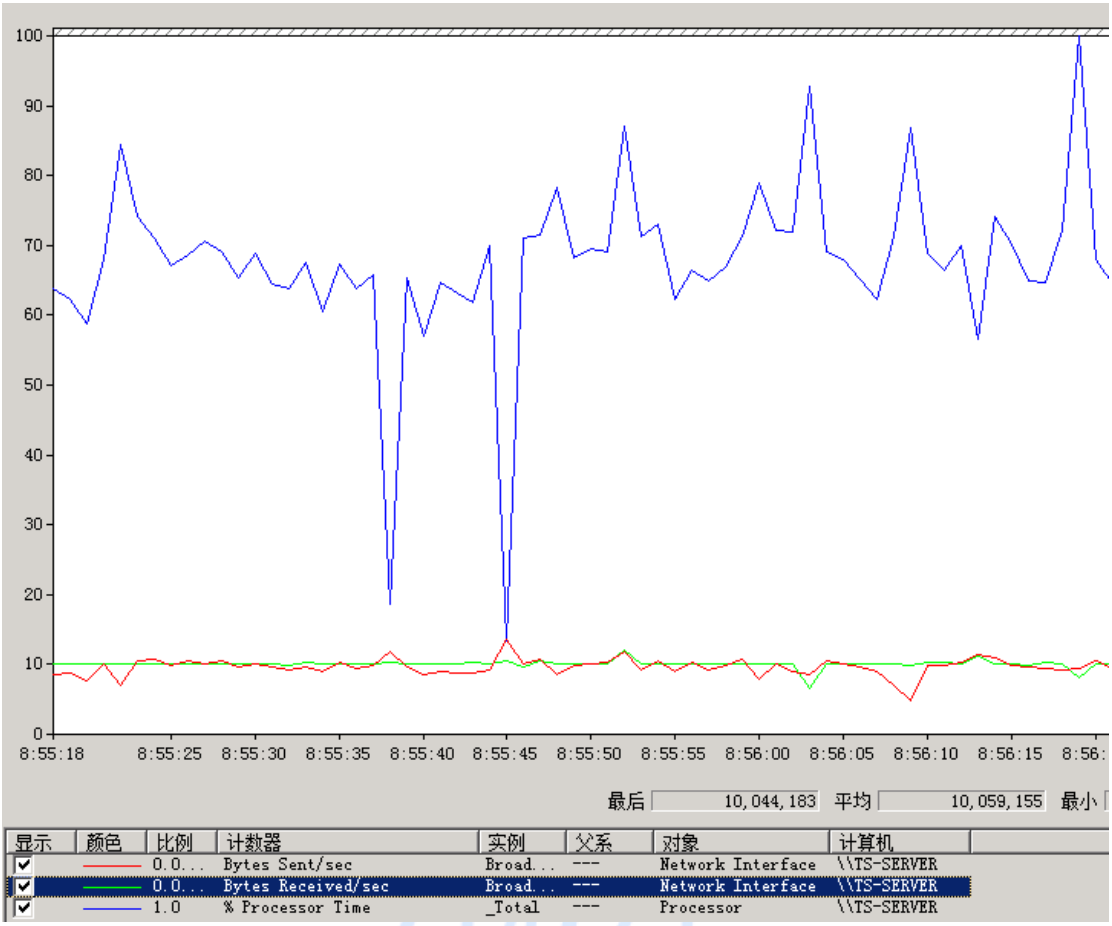
```
ia/external/8253,CS_EXCHANGE_MEDIA,sipp,sipp,192.168.1.253,8253,,,XML,default,PC
MU,8000,64000,PCMU,8000,64000,,TS-Server,,,ACTIVE,Outbound Call,8253,,2b0ba865-a
ee7-4178-a9c5-20d39919e944,,
8e1a2a7c-3228-4c72-9f4b-4c567b96b8aa,inbound,2013-06-15 08:36:10,1371256570,sofi
a/internal/sipp@192.168.1.253:25060,CS_EXECUTE,sipp,sipp,192.168.1.253,82253,bri
dge,sofia/gateway/gw1/82253,XML,default,PCMU,8000,64000,PCMU,8000,64000,,TS-Serv
er,sipp@192.168.1.253,,RINGING,,,,,,
a3552415-3141-473e-a09f-c058018ccfb4,inbound,2013-06-15 08:36:10,1371256570,sofi
a/internal/3006@192.168.1.130,CS_EXECUTE,3006,3006,192.168.1.106,3007,bridge,use
r/3007@192.168.1.130,XML,default,PCMU,8000,64000,PCMU,8000,64000,,TS-Server,3006
@192.168.1.130,,RINGING,,,,,,
d0aab019-7a77-4575-8ff3-2df4b51d2a17,outbound,2013-06-15 08:36:10,1371256570,sof
ia/external/82253,CS_CONSUME_MEDIA,sipp,sipp,192.168.1.253,82253,,,XML,default,P
CMU,8000,64000,PCMU,8000,64000,,TS-Server,,,ACTIVE,Outbound Call,82253,,8e1a2a7c
-3228-4c72-9f4b-4c567b96b8aa,,
332a1c3a-1868-4511-a325-8ef7607b75c2,outbound,2013-06-15 08:36:10,1371256570,sof
ia/internal/sip:3007@192.168.1.106:12100,CS_CONSUME_MEDIA,3006,3006,192.168.1.10
6,3007,,,XML,default,,,,,,,,TS-Server,3007@192.168.1.130,,DOWN,Outbound Call,300
7,,a3552415-3141-473e-a09f-c058018ccfb4,,

2488 total.


freeswitch@TS-Server>
```

稳定运行超过 24 小时。CPU 平均大约占用 65%。内存占用 2.5G 左右，句柄 10 万左右。
如下图：



映像名称	PID	CPU	CPU...	工作设置...	峰值工作设置...	句柄数	线程数	内存(专用...	页面错
360Tray.exe *32	41640	00	0:0...	3,440 K	81,844 K	645	44	2,036 K	
360Tray.exe *32	43172	00	0:0...	4,456 K	79,088 K	664	45	2,412 K	
cmd.exe	47600	00	0:0...	4,588 K	4,620 K	19	1	2,876 K	
conime.exe	10716	00	0:0...	6,388 K	6,388 K	31	1	3,084 K	
csrss.exe	600	00	0:0...	9,972 K	10,020 K	497	13	5,308 K	
csrss.exe	9812	00	0:0...	23,940 K	25,904 K	2,986	13	8,476 K	
csrss.exe	26268	00	0:0...	13,788 K	13,820 K	337	9	4,380 K	
csrss.exe	26276	00	0:0...	10,548 K	10,564 K	98	9	4,948 K	
csrss.exe	57424	00	0:0...	16,224 K	16,256 K	306	9	4,176 K	
dwm.exe	10452	00	0:0...	7,880 K	8,524 K	137	3	3,780 K	
dwm.exe	13716	00	0:0...	7,568 K	8,064 K	137	3	3,584 K	
dwm.exe	56628	00	0:0...	7,812 K	9,756 K	142	3	3,624 K	
explorer.exe	10336	00	0:0...	67,568 K	67,772 K	700	20	41,244 K	
explorer.exe	28788	00	0:0...	56,460 K	56,576 K	590	20	34,408 K	
explorer.exe	61948	00	0:0...	76,884 K	77,196 K	772	19	47,976 K	
FirewallControlPan...	73408	00	0:0...	7,176 K	13,224 K	173	5	2,044 K	
FreeSwitchConsole.exe	76860	64	7:5...	2,003,260 K	2,003,260 K	108,439	2,505	1,994,624 K	
LogonUI.exe	31788	00	0:0...	17,680 K	17,700 K	196	7	10,352 K	
lsass.exe	736	00	0:1...	48,784 K	48,840 K	942	14	40,544 K	
lsm.exe	744	00	0:0...	14,012 K	14,248 K	394	10	9,360 K	
mmc.exe	23060	00	0:1...	238,236 K	304,136 K	507	18	203,896 K	
mmc.exe	54120	01	0:0...	33,732 K	37,864 K	417	15	22,340 K	1,
msdtc.exe	2880	00	0:0...	10,620 K	10,656 K	213	11	5,100 K	
MtxHotPlugService.exe	10332	00	0:0...	5,552 K	5,552 K	14	1	4,028 K	
MtxHotPlugService.exe	32528	00	0:0...	5,560 K	5,560 K	14	1	4,036 K	

网卡 IO 非常大，平均收包达到了 10MByte/S，平均发包也接近 10MByte/S



假如使用 pjsip 进行分机对分机的媒体代理模式持续进行压力测试发现通话过程有内存泄漏，句柄也有泄漏。测试超过 48 小时，内存超过 4G，句柄超过 15 万。在实际环境中使用，发现从 1.2.5 到 1.2.14 的版本都会发生内存泄漏。因此实际部署使用需要使用 64 位的操作系统，才经得起他的泄漏。

假如使用 sipp 的模式测试，FreeSwitch 使用内线拨打网关出口的方式，可以最大接近 3000 线并发，但是出现很多 错误和告警，2000 线左右并发错误告警就很少。只有偶尔出现的无法建立会话的错误：

2013-06-20 16:56:03.452457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-20 16:56:03.452457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-20 17:02:25.132457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-20 17:02:25.132457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-20 17:02:25.132457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-21 00:03:14.792457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-21 00:03:14.912457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-21 05:15:45.692457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-21 05:15:45.692457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-22 10:00:41.912457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-22 10:00:41.912457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-22 11:19:53.072457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-23 08:33:19.472457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-23 09:30:26.912457 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-06-23 09:30:26.912457 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-23 09:30:26.912457 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-23 16:08:02.763000 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-24 04:35:59.743000 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-24 04:35:59.764000 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-24 04:36:00.623000 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-06-24 04:36:00.623000 [CRIT] mod_sofia.c:4831 Error Creating Session

统计结果是测试 100 小时，总共出现 23 次错误。

测试 2000 线左右并发，测试 100 小时，没有发现内存泄漏，内存最高 2G，句柄不超过 10 万，如下图：

映像名称	PID	CPU	CPU 时间	工作设置...	峰值工作设置...	句柄数	线程数	内存(专用...	页面错...	基本...	工
360Tray.exe *32	3224	00	0:01:49	3,928 K	81,440 K	671	46	2,268 K		标准	
360Tray.exe *32	4344	00	0:01:06	3,484 K	81,132 K	576	45	2,076 K		标准	
cmd.exe	9580	00	0:00:00	4,560 K	4,592 K	19	1	2,864 K		标准	
codeblocks.exe *32	6604	00	1:18:24	108,324 K	108,336 K	273	9	83,044 K		标准	
codeblocks.exe *32	15312	00	0:56:24	120,848 K	122,924 K	381	9	90,608 K		标准	
conime.exe	2924	00	0:00:00	6,692 K	6,692 K	31	1	3,128 K		标准	
conime.exe	4324	00	0:00:00	13,552 K	17,704 K	114	1	6,832 K		标准	
csrss.exe	600	00	0:00:11	10,204 K	10,240 K	481	13	5,572 K		标准	
csrss.exe	644	00	0:00:09	10,420 K	10,428 K	98	8	4,844 K		标准	
csrss.exe	684	00	0:00:55	19,256 K	20,008 K	2,431	12	7,588 K		标准	
csrss.exe	1304	00	0:00:15	20,076 K	21,124 K	409	11	7,284 K		标准	
csrss.exe	8272	02	0:00:00	7,336 K	7,340 K	70	8	3,264 K	3,931	标准	
dwm.exe	404	00	0:00:00	8,276 K	9,640 K	145	3	3,752 K		标准	
dwm.exe	1932	00	0:00:00	7,864 K	8,500 K	136	3	3,784 K		标准	
explorer.exe	3084	00	0:16:38	85,676 K	85,936 K	871	26	58,680 K	10	标准	
explorer.exe	4176	00	0:07:52	135,624 K	138,040 K	801	23	98,588 K	6	标准	
FreeSwitchConsole.exe	10480	60	1258:26:26	1,653,428 K	1,997,324 K	84,809	1,935	1,644,176 K		高	
IPMSG.exe *32	14612	00	0:00:03	15,896 K	16,124 K	246	3	5,164 K		标准	

假如是通话全程录音模式，

FreeSwitch 的 cpu 使用大大增加。

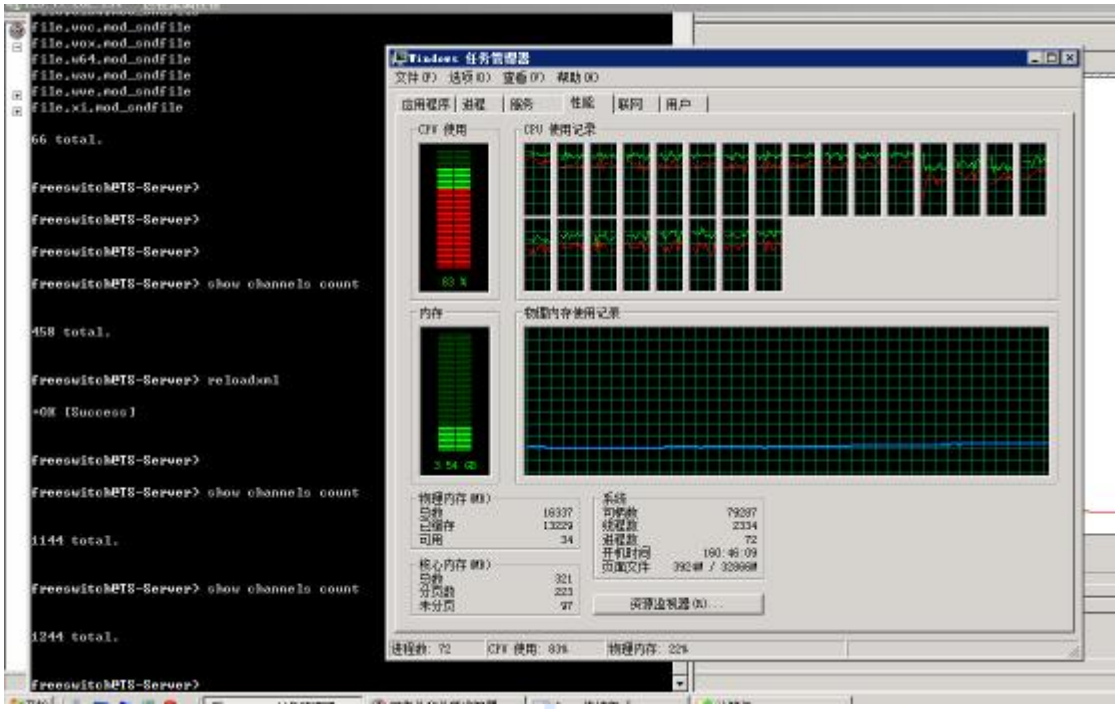
使用下面的录音配置，录制为 alaw 格式的语音：

```
<action application="set" data="record_sample_rate=8000"/>
<action application="export" data="RECORD_STEREO=false"/>
  <action application="set" data="hangup_after_bridge=true"/>
  <action application="set" data="continue_on_fail=true"/>
<action application="export"
data="execute_on_answer=record_session
d:/data/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_${destination_number}_${caller_id_number}_${uuid}.alaw"
/>
  <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
```

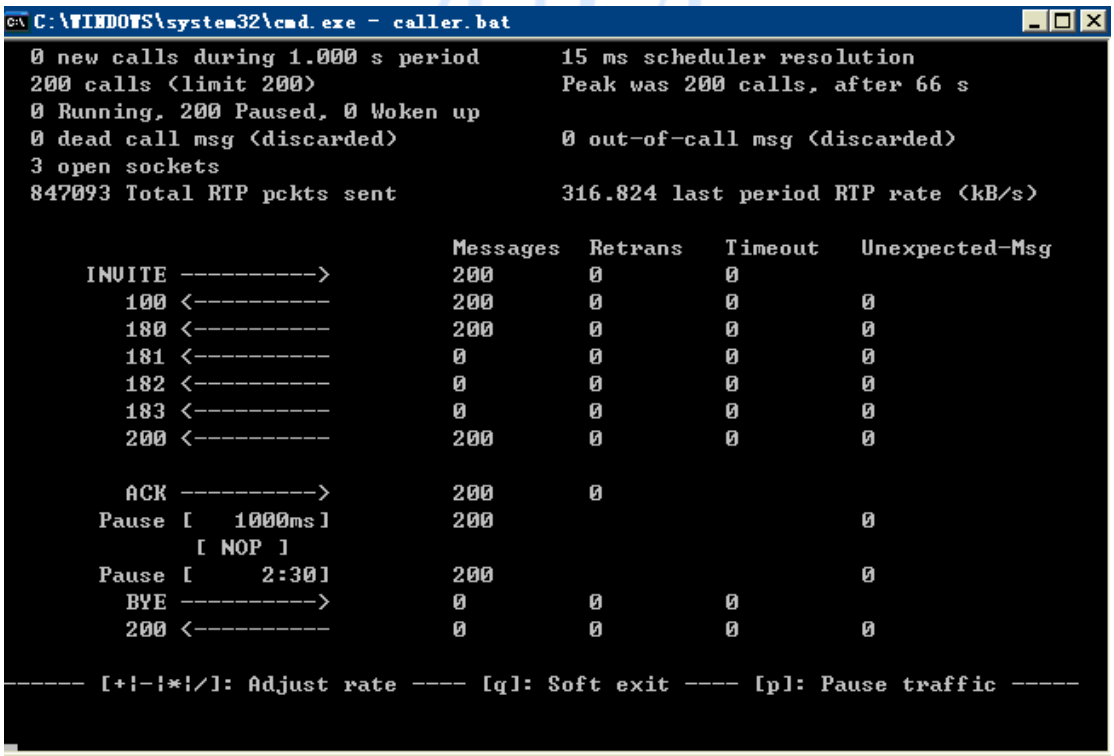
同样配置的服务器，总共大约 1244 线并发,其中 344 线没有录音，900 线全程录音，

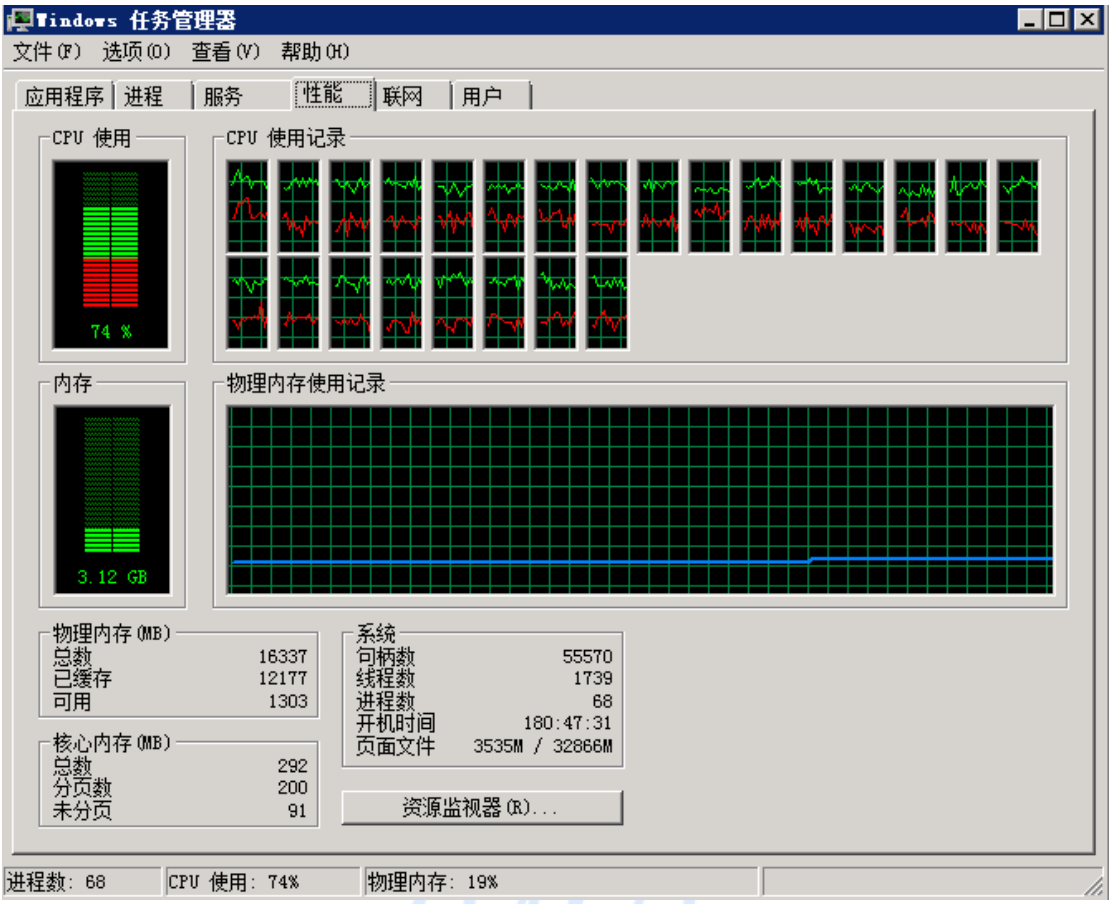
Cpu 使用平均达到了 83%，远远超过之前的 2488 线的并发通话没有录音的 cpu 使用平均 65%的情况。

如下图：



假如使用 mp3 格式录音，因为要对语音进行 mp3 编码 cpu 使用就更加恐怖。200 线 cpu 使用就已经平均达到 74%，对比通话的 2000 线以上的并发，整整并发的容量缩小了一个数量级。如下图：：





结论:在普通对话模式下，E5-2640*2cpu 配置的服务器可以支持 2000 线并发通话。
在全程录音的模式下，E5-2640*2cpu 配置的服务器，alaw 或者 ulaw 格式录音，单机只能支持 1000 线并发，是通话模式的 1/2。
假如是 Mp3 格式录音，因为要对语音进行 mp3 编码,单机只能支持 200 线并发,是通话模式的 1/10。
曾经怀疑是硬盘 IO 引发的问题，于是找以另外一个配置 CPU 3612QM，6G 内存（使用 Primo Ramdisk Ultimate Edition 软件划出 2G 作为内存硬盘）的机器继续进行测试：

在没有录音的情况下，900 线并发运行使用平均 50%的 cpu

在全程录音的情况下，录音录制到 RAMDISK 的内存盘上， 600 线并发，就已经占用 60%的 cpu，假如是录制为 mp3 的语音， 100 线就已经达到 50%。

因为 RAMDISK 的内存盘的硬盘 IO 是 GB/S 级别，不会给录音带来阻塞，排除了 CPU 暴涨是由于硬盘 IO 导致的问题。

另外，本人测试使用 mp3 格式的压缩率为 3:1，建议除非是特别考虑 i/o 或者硬盘容量的情况下，不要使用 mp3 格式的录音。因为后续使用 winrar 之类的做离线压缩，压缩率也可以达到 2:1.

116. 如何把 FreeSwitch 默认使用的 SQLLite 迁移到 MySql 上以提高性能？

FreeSwitch 默认使用的工作数据库是 SqlLite，在 \db 目录下：

名称	修改日期	类型	大小
call_limit.db	2013/5/23 19:46	Data Base File	14 KB
core.db	2013/7/2 13:18	Data Base File	842 KB
fifo.db	2013/7/4 22:47	Data Base File	5 KB
sofia_reg_external.db	2013/5/23 19:49	Data Base File	90 KB
sofia_reg_internal.db	2013/7/2 13:18	Data Base File	343 KB
sofia_reg_internal-ipv6.db	2013/5/23 19:49	Data Base File	90 KB
voicemail_default.db	2013/5/23 19:46	Data Base File	16 KB

在压力测试的时候，或者高并发使用的时候,可能会出现数据库LOCK的问题，类似下面：

```
[ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [database is locked]
BEGIN EXCLUSIVE
[CRIT] switch_core_sqldb.c:1712 ERROR [database is locked]
[ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [cannot commit - no transaction is active]
COMMIT
```

为了提高性能，减小数据库锁的问题，可以考虑把 FreeSwitch 工作的数据库迁移到 mysql 数据库里面。

Linux 下的迁移步骤请参照：

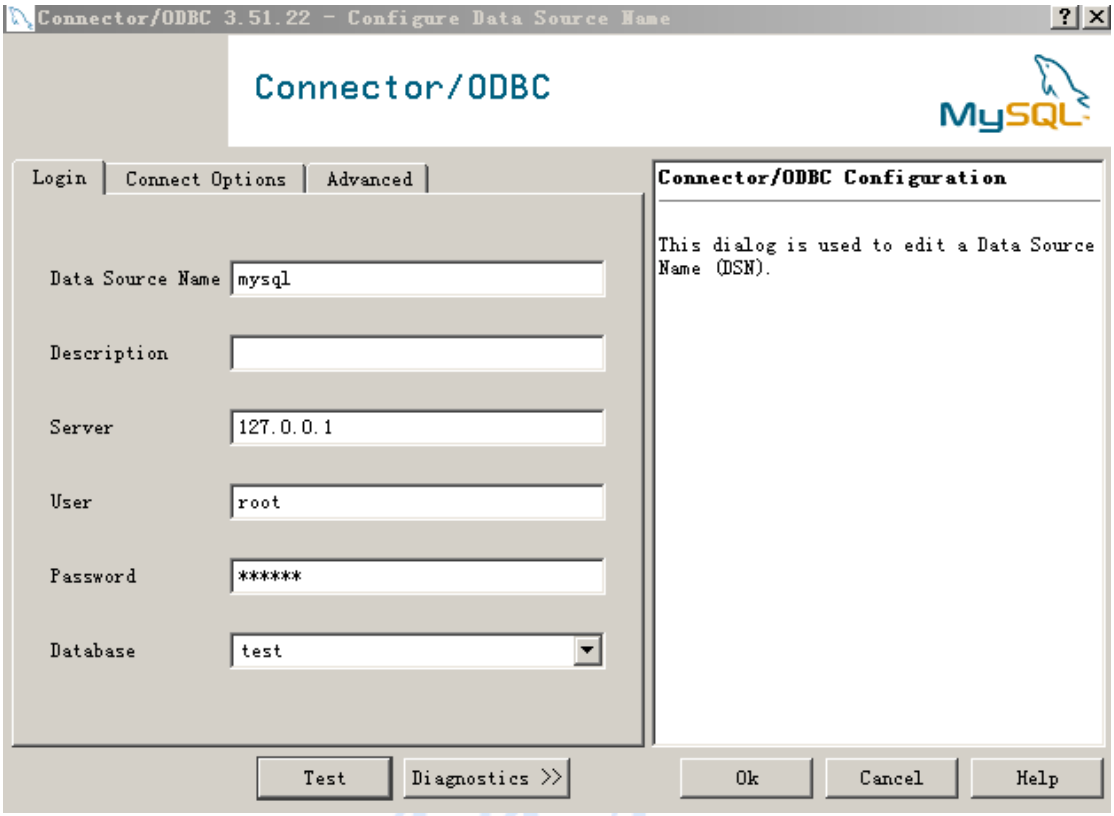
http://wiki.FreeSwitch.org/wiki/Using_ODBC_in_the_core

按照上面做就可以了，

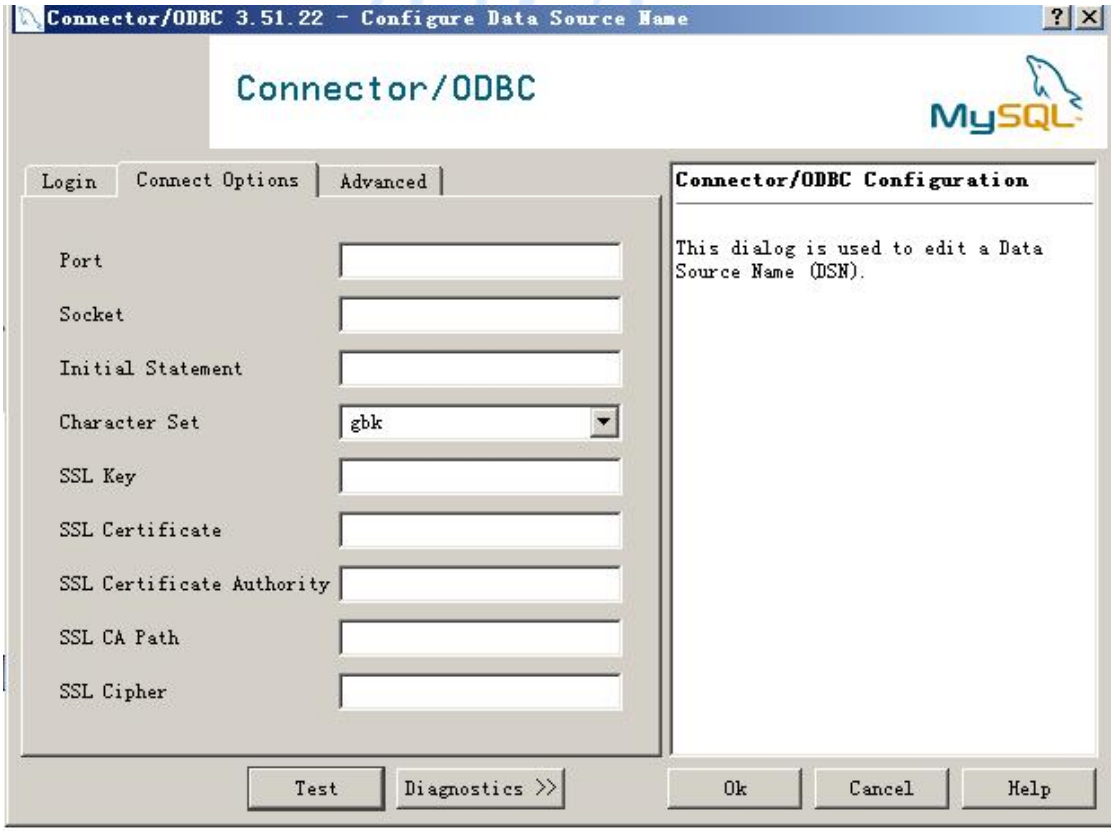
注意 ubuntu 安装 unixODBC 的路径和说明里面的不一样的，需要自己修改一下。

Windows 操作系统下我的迁移步骤如下：

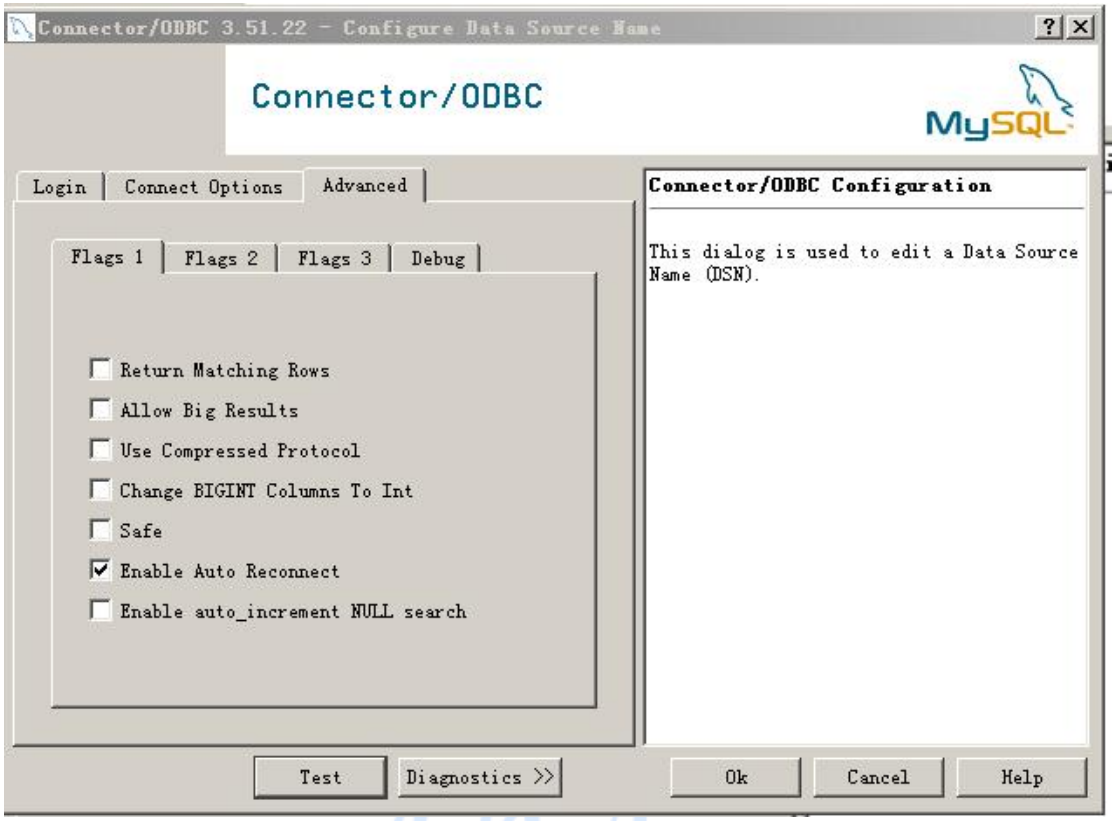
1. 安装 mysql 数据库, mysql-5.5.29-win32 版本，配置 root 的密码为 123456，字符集为 gbk。
2. 安装 mysql odbc 驱动， mysql-connector-odbc-3.51.22-win32，
3. 配置 mysql 的 odbc，dsn 名叫做 mysql 使用默认的 test 数据库



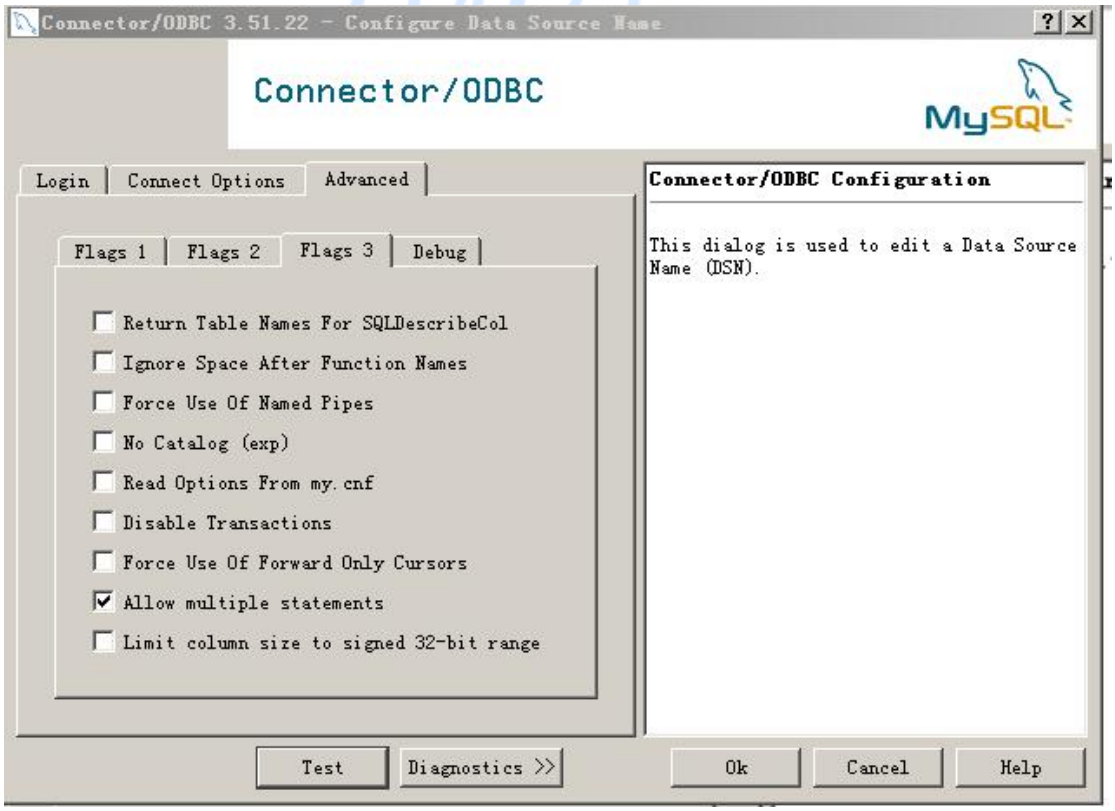
选择 gbk 字符集



选择自动重新连接

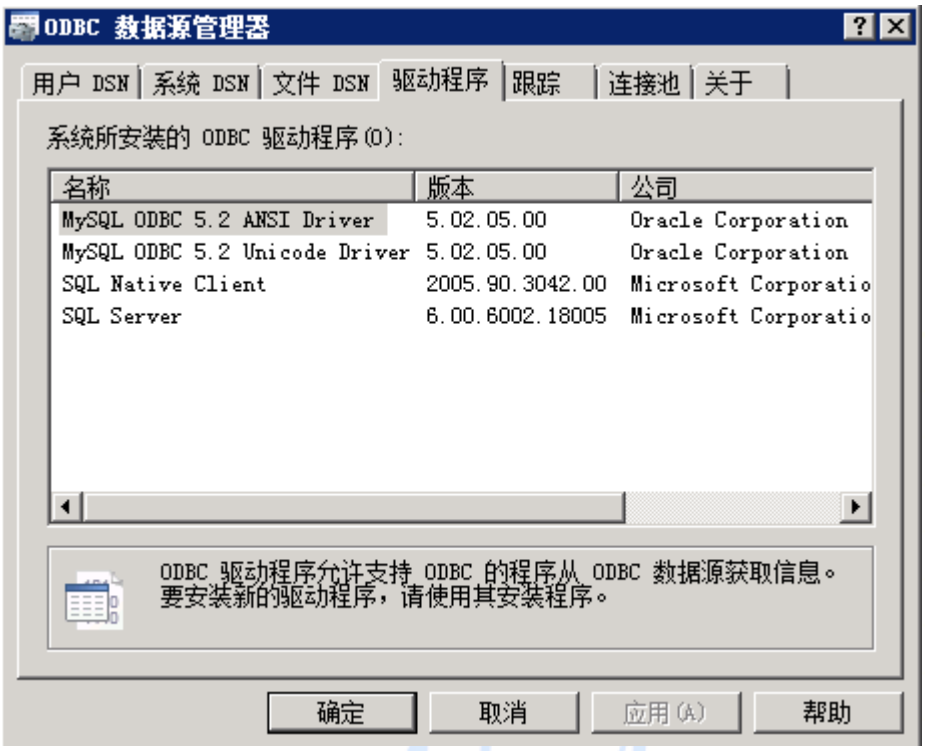


选择允许多条语句执行： ALLOW MULTI STATEMENTS 要勾上，



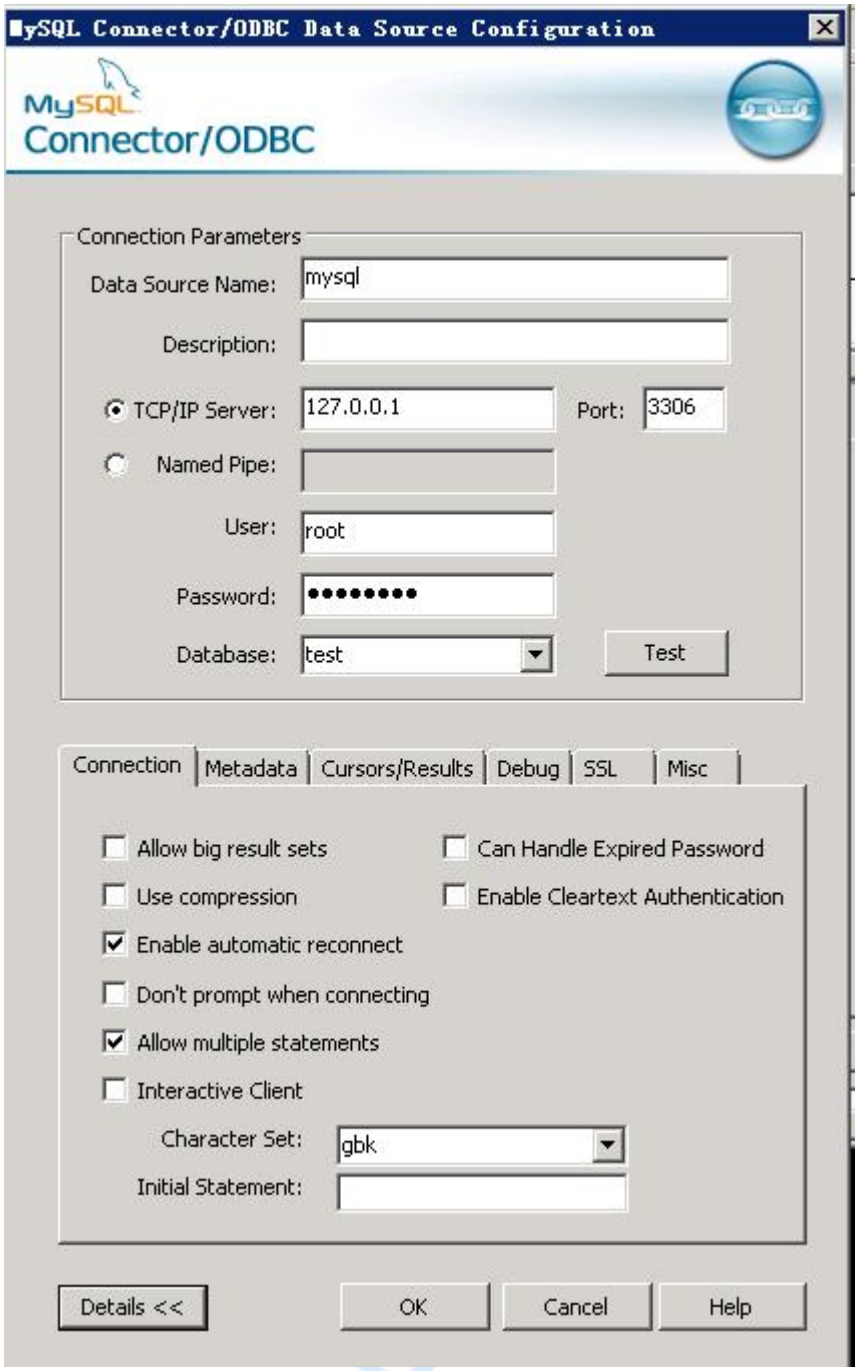
假如是 64 位 WINDOWS 操作系统的请安装 mysql-connector-odbc-winx64-5.2.5（当然也可以使用 32 位的驱动，不过配置有点不一样，参见下面的说明部分）， 然后在控制面板里面进行配置如下：

选择 ansi driver。



说明：假如你是在 64 位的 windows 操作系统安装的却是 32 位的 ODBC 驱动，
那么不能使用控制面板里面的 ODBC 配置来配置，要手工打开 64 位目录下的配置程序进行配置：
C:\Windows\SysWOW64\odbcad32.exe

然后配置



4. 配置 internal.xml，把里面的注释放开

```
<param name="odbc-dsn" value="mysql:root:123456"/>
```

5. 配置 external.xml 里面的参数，默认没有这个，自己添加 到 <settings>。。。 </settings>里面

```
<param name="odbc-dsn" value="mysql:root:123456"/>
```

6. 配置 switch.conf.xml，把里面的注释放开

```
<param name="core-db-dsn" value="mysql:root:123456" />
```

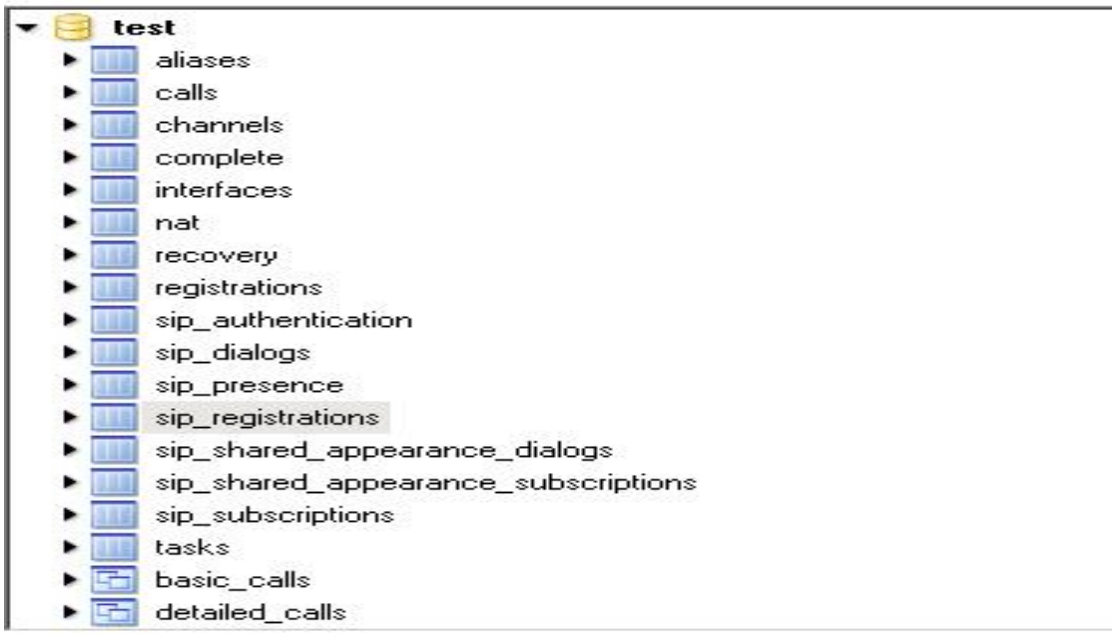
7. 假如有需要还要迁移配置 fifo.conf.xml 的和 voicemail.conf.xml 里面的 db，
   <param name="odbc-dsn" value="dsn:user:pass"/>
   偶没有使用到这两个，因此不需要。

8. 重启 FreeSwitch，
   因为 mysql 里面的 FreeSwitch 工作的表不存在，前面几次重启会有告警，
   不过 FreeSwitch 会自动建立工作的表，不用我们操心，假如有看到告警提示表不存在之类的之后，多重启几次:

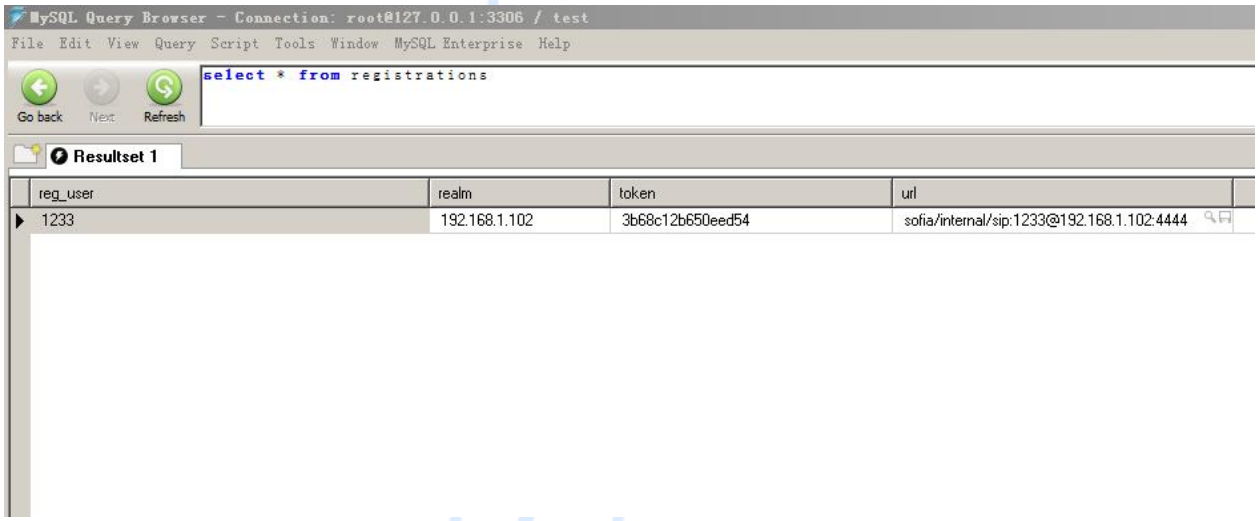
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1192 SQL ERR [no such table: sip_registrations]
   [drop table sip_registrations]
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1185 SQL ERR [no such table: sip_registrations]
   [delete from sip_registrations where (sub_host is null or contact like '%TCP%' or status like '%TCP%' or status like '%TLS%') and hostname='yhy-PC32' and network_ip like '%' and network_port like '%' and sip_username like '%' and mwi_user like '%' and mwi_host like '%' and orig_server_host like '%' and orig_hostname like '%']
   Auto Generating Table!
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1192 SQL ERR [no such table: sip_registrations]
   [drop table sip_registrations]
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1185 SQL ERR [no such table: sip_subscriptions]
   [delete from sip_subscriptions where hostname='yhy-PC32' and full_to='XXX']
   Auto Generating Table!
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1192 SQL ERR [no such table: sip_subscriptions]
   [DROP TABLE sip_subscriptions]
   2013-05-23 19:45:01.827621 [DEBUG] switch_core_sqldb.c:1185 SQL ERR [no such table: sip_subscriptions]

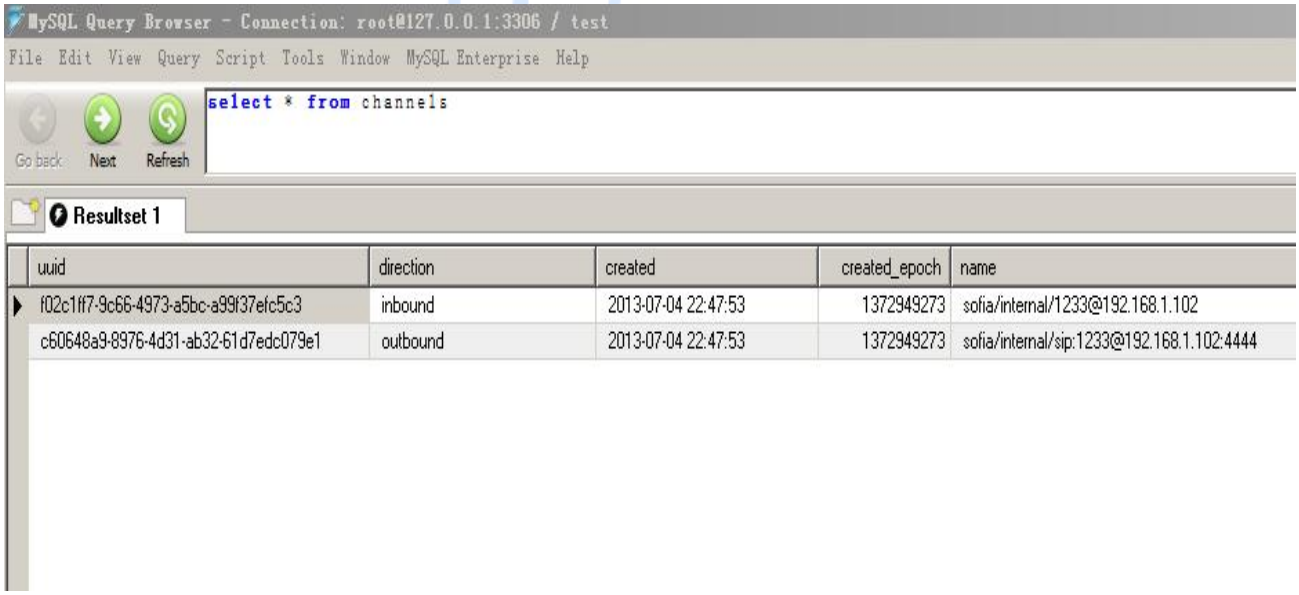
   退出重启，还有告警
   2013-05-23 19:45:03.855336 [DEBUG] switch_core_sqldb.c:1185 SQL ERR [no such table: fifo_bridge]
   [delete from fifo_bridge]
   Auto Generating Table!
   2013-05-23 19:45:03.855336 [DEBUG] switch_core_sqldb.c:1192 SQL ERR [no such table: fifo_bridge]
   [drop table fifo_bridge]
   2013-05-23 19:45:03.857336 [DEBUG] switch_core_sqldb.c:1185 SQL ERR [no such table: fifo_callers]
   [delete from fifo_callers]
   Auto Generating Table!

   再退出，重启。
   注册软电话上来，拨打测试。
   可能使用过程还会有告警。这是因为呼叫功能工作的表不存在，FreeSwitch 也会自动建立。重启告警就消失。
   正常使用之后，FreeSwitch 会在 mysql 里面建立下面的表和视图：



可以通过 mysql 的客户端工具查询表内容，比如查询注册用户表和通道表：





迁移之后重新做压力测试：结果是同样的配置的机器，支持的并发通话数目没有明显提高，但是错误告警减小很多。特别是不会出现下面的数据库的操作错误告警：

```
2013-06-15 08:36:42.746089 [ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [database is locked]
BEGIN EXCLUSIVE
2013-06-15 08:36:42.746089 [CRIT] switch_core_sqldb.c:1712 ERROR [database is locked]
2013-06-15 08:36:42.746089 [ERR] switch_core_sqldb.c:579 NATIVE SQL ERR [cannot commit - no transaction is active]
COMMIT
```

不过运行过程中，偶尔还是会出现：

```
 FreeSwitch@TS-Server> 2013-07-13 00:11:43.425200 [CRIT] mod_sofia.c:4831 Error C
reating Session
2013-07-13 01:57:07.428200 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-07-13 01:57:07.445200 [CRIT] mod_sofia.c:4831 Error Creating Session
2013-07-13 07:49:22.145200 [CRIT] mod_sofia.c:4831 Error Creating Session
```

的错误，但是很少。

假如还要再次提升数据库操作的性能，还有其它的办法可以再次提高性能：

可以考虑把这些工作表改为 mysql 的内存表。

由于 FreeSwitch 这些数据库是用来跑临时数据的，所以不需要保存的。

所以可以考虑把本来是 Innodb 的表都改成 mysql 的内存表了。

大家可以自己尝试一下，把那些表脚本导出之后，参照下面的语句

修改脚本，以 sip_registrations 为例：

```
DROP TABLE IF EXISTS `test`.`sip_registrations`;
CREATE TABLE   `test`.`sip_registrations` (
  `call_id` varchar(255) DEFAULT NULL,
  `sip_user` varchar(255) DEFAULT NULL,
  `sip_host` varchar(255) DEFAULT NULL,
  `presence_hosts` varchar(255) DEFAULT NULL,
  `contact` varchar(1024) DEFAULT NULL,
  `status` varchar(255) DEFAULT NULL,
  `rpid` varchar(255) DEFAULT NULL,
  `expires` int(11) DEFAULT NULL,
  `user_agent` varchar(255) DEFAULT NULL,
  `server_user` varchar(255) DEFAULT NULL,
  `server_host` varchar(255) DEFAULT NULL,
  `profile_name` varchar(255) DEFAULT NULL,
  `hostname` varchar(255) DEFAULT NULL,
  `network_ip` varchar(255) DEFAULT NULL,
  `network_port` varchar(6) DEFAULT NULL,
  `sip_username` varchar(255) DEFAULT NULL,
  `sip_realm` varchar(255) DEFAULT NULL,
  `mwi_user` varchar(255) DEFAULT NULL,
  `mwi_host` varchar(255) DEFAULT NULL,
  `orig_server_host` varchar(255) DEFAULT NULL,
  `orig_hostname` varchar(255) DEFAULT NULL,
```

```
  `sub_host` varchar(255) DEFAULT NULL,
  KEY `sr_call_id` (`call_id`),
  KEY `sr_sip_user` (`sip_user`),
  KEY `sr_sip_host` (`sip_host`),
  KEY `sr_sub_host` (`sub_host`),
  KEY `sr_mwi_user` (`mwi_user`),
  KEY `sr_mwi_host` (`mwi_host`),
  KEY `sr_profile_name` (`profile_name`),
  KEY `sr_presence_hosts` (`presence_hosts`),
  KEY `sr_contact` (`contact`(383)),
  KEY `sr_expires` (`expires`),
  KEY `sr_hostname` (`hostname`),
  KEY `sr_status` (`status`),
  KEY `sr_network_ip` (`network_ip`),
  KEY `sr_network_port` (`network_port`),
  KEY `sr_sip_username` (`sip_username`),
  KEY `sr_sip_realm` (`sip_realm`),
  KEY `sr_orig_server_host` (`orig_server_host`),
  KEY `sr_orig_hostname` (`orig_hostname`)
) ENGINE= MEMORY    DEFAULT CHARSET=gbk;
```

注意上面的 MEMORY，就是要修改的地方，表示是内存表。

内存表与实体表的区别：内存表不保存数据到硬盘，于是内存表超级快。

假如机器重启表里面的数据就没有了。

但是内存表的建表的语句是保存在硬盘的，mysql 启动的时候会重新在内存里面建立这些内存表。

由于 FreeSwitch 的工作表都是临时的数据，因此不影响使用。

117. 如何使用 PJSIP 对 FreeSwitch 的 IVR 进行压力测试？

上面说过 PJSIP 稍微修改一下就可以对 FreeSwitch 进行压力测试。

FreeSwitch IVR 的压力测试跟 FreeSwitch 单纯的分机呼叫分机或者分机转出到网关的压力测试不大一样。

IVR 的通常需要测试一些关键的路径，比如按 0 转人工台，按 1 参加会议啥的。

跟 Sipp 比较，PJSIP 的优势是可以播放 WAV 文件， 这样的好处是可以很容易地制作 IVR 压力测试的用户场景，而 Sipp 只能播放抓包的 cap 文件，制作用户场景就比较麻烦。

使用 PJSIP 对 IVR 进行压力测试需要下面的步骤：

1. 制作客户端软电话的自动呼叫器：

   修改 PJSIP，使它能自动拨出到 FreeSwitch 上的某个号码上。

以 pjsip 1.16 版本为例进行修改，其它版本基本类似。

在 默认的 pjsip 的工程编译通过之后，

首先 修改 pjproject-1.16\pjlib\include\pj\config_site.h 内容如下：

```
#define PJSUA_MAX_CALLS         256   //最大通话数
#define PJ_LOG_MAX_LEVEL        1   //日志级别，级别越高日志越详细
#define PJMEDIA_HAS_L16_CODEC           0   //禁用 L16
#define PJMEDIA_HAS_G722_CODEC          0   //禁用 722
```

```
#define PJMEDIA_HAS_SPEEX_CODEC          0   //禁用 SPEEX
#define PJSIP_MAX_TRANSPORTS         512
#define PJ_IOQUEUE_MAX_HANDLES           512
```

然后修改 pjsua 工程里面的源码:

先修改 pjsua_app.c 里面的 console_app_main 函数

```
void console_app_main(const pj_str_t *uri_to_call)
{
    char menuin[32];
    char buf[128],tmp2[128];
    char text[128];
    int i, count;
    char *uri;
    pj_str_t tmp;
    struct input_result result;
    pjsua_msg_data msg_data;
    pjsua_call_info call_info;
    pjsua_acc_info acc_info;
    /* If user specifies URI to call, then call the URI */
    if (uri_to_call->slen) {
       pjsua_call_make_call( current_acc, uri_to_call, 0, NULL, NULL, NULL);
    }
  GetPrivateProfileString("SET", "maxcall", "0",tmp2,32, "./set.cfg");
  maxcalls=atoi(tmp2);
  GetPrivateProfileString("SET", "sipnum", "1003",sipnum,32,"./set.cfg");
  GetPrivateProfileString("SET", "remoteip", "192.168.1.101",remoteip,32,"./set.cfg");
  GetPrivateProfileString("SET", "remotesipport", "5060",remotesipport,32,"./set.cfg");
  GetPrivateProfileString("SET", "delay", "10000",tmp2,32, "./set.cfg");
  Interval =atoi(tmp2);
  if(maxcalls>0) CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)MakeCallThread,NULL,0, NULL);
 keystroke_help();
  setcodec(1);
。。。
```

增加斜体部分。Setcodec 函数是设置编码的优先级，下面会介绍。

然后在 pjsua_app.c 里面增加外拨的线程：

```
int maxcalls=1,Interval=10000;
char remoteip[128],remotesipport[128],sipnum[128];
static pj_thread_desc handler_desc1;
int makecallcount=0;
void mymakecall(char    buf[128]);
int stop=0;
int MakeCallThread()
{
  char buf[128];
  pj_thread_t *thread;
```

```
  pj_thread_register("MakeCallThread", handler_desc1, &thread);
  while(!stop)
  {
        Sleep(100);
        if(mymakecallcount<maxcalls)
        {
              sprintf(buf,"sip:%s@%s:%s",sipnum,remoteip,remotesipport);
              makecall(buf);
        }
        Sleep(Interval);
  }
  return 0;
}
```

然后在 pjsua_app.c 里面增加外拨的函数：

```
void mymakecall(char    buf[128])
{
      char *uri;
      pj_str_t tmp;
      struct input_result result;
  pjsua_call_get_count();
  uri = NULL;
  my_ui_input_url("Make call", buf, sizeof(buf), &result);
  if (result.nb_result != NO_NB) {
        if (result.nb_result == -1 || result.nb_result == 0) {
              //puts("You can't do that with make call!");
              return;
        } else {
              pjsua_buddy_info binfo;
              pjsua_buddy_get_info(result.nb_result-1, &binfo);
              uri = binfo.uri.ptr;
        }
  } else if (result.uri_result) {
        uri = result.uri_result;
  }
  tmp = pj_str(uri);
  pjsua_call_make_call( current_acc, &tmp, 0, NULL, NULL, NULL);
  printf("makecall makecallcount=%d\n",makecallcount);
}
```

然后在 pjsua_app.c 里面增加 my_ui_input_url 函数：

```
 static void my_ui_input_url(const char *title, char *buf, int len,
                  struct input_result *result)
{
      result->nb_result = NO_NB;
      result->uri_result = NULL;
```

```
        print_buddy_list();
        len = strlen(buf);
        /* Left trim */
        while (pj_isspace(*buf)) {
           ++buf;
           --len;
        }
        /* Remove trailing newlines */
        while (len && (buf[len-1] == '\r' || buf[len-1] == '\n')) buf[--len] = '\0';
        if (len == 0 || buf[0]=='q')        return;
    if (pj_isdigit(*buf) || *buf=='-') {
     int i;
     if (*buf=='-')
           i = 1;
     else
           i = 0;
     for (; i<len; ++i) {
           if (!pj_isdigit(buf[i])) {
            //puts("Invalid input");
            return;
           }
     }
     result->nb_result = my_atoi(buf);
     if (result->nb_result >= 0 &&
           result->nb_result <= (int)pjsua_get_buddy_count())
     {
           return;
     }
     if (result->nb_result == -1)
           return;
     result->nb_result = NO_NB;
     return;
        } else {
     pj_status_t status;
     if ((status=pjsua_verify_sip_url(buf)) != PJ_SUCCESS) {
           //pjsua_perror(THIS_FILE, "Invalid URL", status);
           return;
     }
     result->uri_result = buf;
        }
}
```

最后修改 on_call_state 函数，对当前的并发呼叫个数进行计数：

```
static void on_call_state(pjsua_call_id call_id, pjsip_event *e)
{
```

```
        pjsua_call_info call_info;
        PJ_UNUSED_ARG(e);
        pjsua_call_get_info(call_id, &call_info);
        if (call_info.state == PJSIP_INV_STATE_DISCONNECTED) {
        。。。
           makecallcount--;
        } else
        {
            。。。
                    if(strcmp(call_info.state_text.ptr, "CALLING") == 0)        makecallcount++;
}
                if (current_call==PJSUA_INVALID_ID)
                    current_call = call_id;
        }
```

然后增加下面这些跟设置编码器优先级有关的函数

```
static void my_manage_codec_prio(char* input)
{
        char *codec, *prio;
        pj_str_t id;
        int new_prio;
        pj_status_t status;
        codec = strtok(input, " \t\r\n");
        prio = strtok(NULL, " \r\n");
        if (!codec || !prio) {
                return;
        }
        new_prio = atoi(prio);
        if (new_prio < 0)
           new_prio = 0;
        else if (new_prio > PJMEDIA_CODEC_PRIO_HIGHEST)
           new_prio = PJMEDIA_CODEC_PRIO_HIGHEST;
        status = pjsua_codec_set_priority(pj_cstr(&id, codec),
                                (pj_uint8_t)new_prio);
}
void setcodec(int codec)
{
  char input[32];
  pjsua_codec_info c[32];
   unsigned i, count = PJ_ARRAY_SIZE(c);
  switch(codec)
  {
  case 0: //G711 优先
        strcpy(input, "PCMA/8000 200");
        my_manage_codec_prio(input);
```

```
            strcpy(input, "PCMU/8000 200");
            my_manage_codec_prio(input);
            break;
    case 1: //iLBC 优先
            strcpy(input, "iLBC/8000 200");
            my_manage_codec_prio(input);
            break;
    case 2: //SILK 优先 PJSIP VER2.1 才支持，1.6 不支持
            strcpy(input, "SILK/8000 200");
            my_manage_codec_prio(input);
            break;
    default:
            strcpy(input, "PCMA/8000 200");
            my_manage_codec_prio(input);
            strcpy(input, "PCMU/8000 200");
            my_manage_codec_prio(input);
    }
    printf( "List of codecs:");
        pjsua_enum_codecs(c, &count);
      for (i=0; i<count; ++i)
    {
            if(c[i].priority > 0)
                printf("    %d\t%.*s\n", c[i].priority, (int)c[i].codec_id.slen,
                c[i].codec_id.ptr);
        }
  }
```

然后就可以编译了，记得选择 release 版本进行编译，这样 pjsua_vc6.exe 就准备好了。

然后准备 set.cfg 配置文件如下：

```
[SET]
#呼叫间隔，单位毫秒，1000 表示间隔 1000 毫秒假如允许呼叫的话就呼叫一次。
delay=1000
#外拨呼叫的最大呼叫个数，不主动外拨=0，比如要允许最大 10，就设置为 10
maxcall=10
#外拨呼叫的被叫号码
sipnum=8000
#外拨呼叫 FS IP 地址
remoteip=192.168.1.102
#外拨呼叫 FS 端口
remotesipport=5060
```

然后写测试的呼叫批处理文件 test.bat，因为很多需要参数，写 bat 可以一劳永逸。

test.bat 内容:

```
pjsua_vc6.exe --id sip:1001@192.168.1.102 --registrar sip:192.168.1.102 --realm 192.168.1.102 --username 1001 --local-port=11001 --rtp-port=17011 --auto-play --play-file 1001.wav --password 1234 --reg-timeout 60 --auto-answer 200
```

--max-calls 200 --thread-cnt 4 --duration=180 --no-tcp --null-audio

具体的参数可以 查询上面的 PJSIP 如何编译的介绍问题。

这里重点 提一下下面两个参数：

--max-calls 200 表示本进程最大 200 线呼叫。这个数目不能超过 PJSUA_MAX_CALLS

--duration=180 表示 通话时间是 180 秒。这个是跟 1001.wav 文件相关的，

假如 1001.wav 文件小于 180 秒，那么会循环播放，因此，为了测试不会神鬼乱舞，

建议 1001.wav 的语音时间长度跟 duration 的配置一致，或者长一些。

最后 使用 cooledit 2000 制作 1001.wav 文件。这个 1001.wav 代表用户场景：

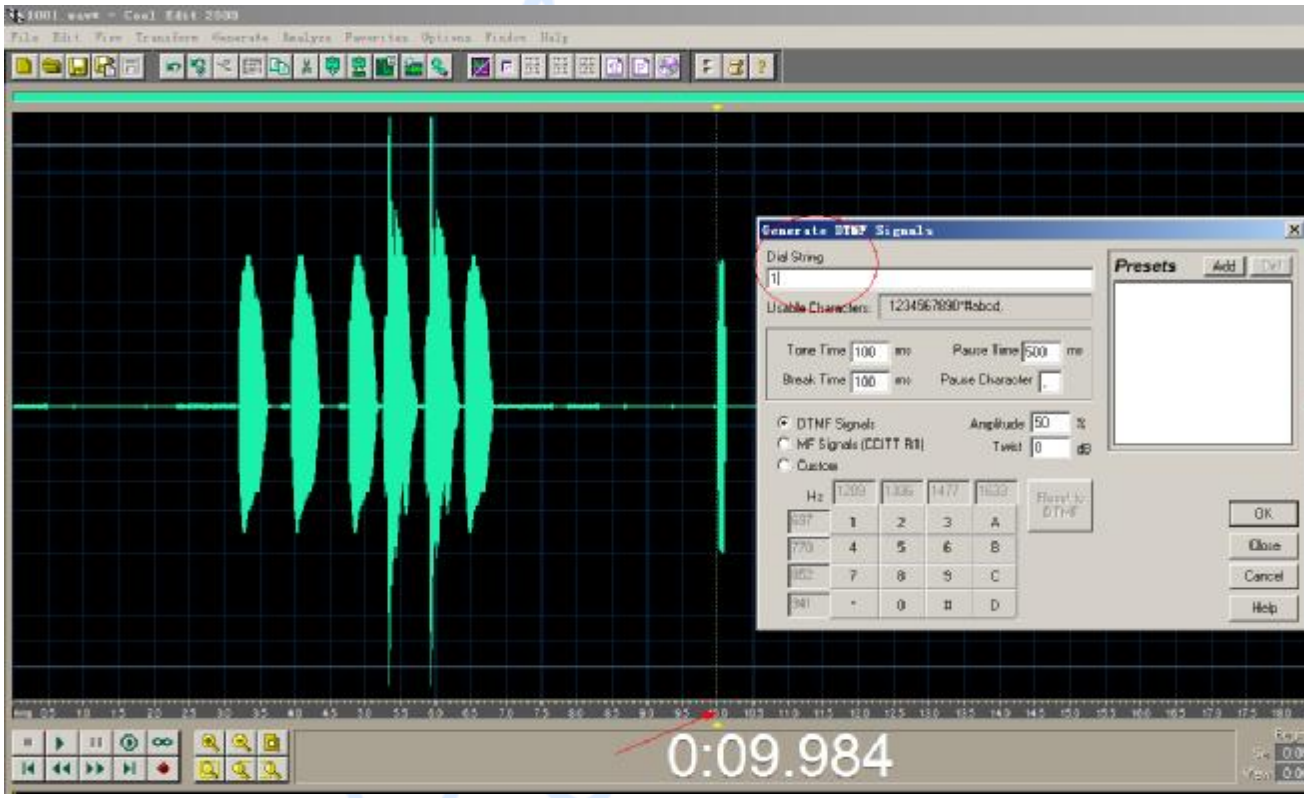
比如要模拟用户播打系统，接通之后第 10 秒按 1 键，第 15 秒按 2，第 20 秒按 3。。。

可以如下操作：

先使用 cooledite 2000 录音语音比如 180 秒或者 30 秒。不够可以使用静音凑

然后移动到第 10 秒的地方，然后选择【Generate】菜单【DTMF Signals...】子菜单

然后依照下面输入 1



然后就会产生按键 1 的 DTMF 按键声音信号。第 15 秒按 2，第 20 秒按 3 依照此方法类似操作。然后保存 wav 文件。 这样客户端的工作就算准备好了。

注意：呼叫的时候注意使用 alaw 或者 ulaw 编码器。这个可以在客户端上指定或者在 FreeSwitch 服务器配置都可以。否则带内的 DTMF 可能由于压缩的原因不工作。

2. 修改 FreeSwitch 服务器配置

这种在语音里面带 DTMF 按键声音的称为带内 inband 按键。

FreeSwitch 默认是支持 RFC2833 的 DTMF 检测不支持带内 DTMF 检测，

需要如下修改：

a.修改 conf\sip_profiles\internal.xml 里面参数：

```
<param name="dtmf-type" value="inband"/>
```

b.修改 拨号计划 conf\dialplan\default.xml 里面拨号参数：

```
<extension name="Local_Extension2">
```

```
<condition field="destination_number" expression="^([0-9]\d+)$">
                <action application="export" data="dialed_extension=$1"/>
                 <action application="start_dtmf" />
                <action application="set" data="call_timeout=30"/>
                 <action application="set" data="record_sample_rate=8000"/>
                 <action application="export" data="RECORD_STEREO=false"/>
                <action application="set" data="hangup_after_bridge=true"/>
                <action application="set" data="continue_on_fail=true"/>
                 <action application="socket" data="127.0.0.1:8084 async full"/>
            </condition>
    </extension>
```

注意上面的斜体部分。重启 FreeSwitch 之后，启动 ESL 的 IVR 连接到 FreeSwitch 就可以使用 test.bat 进行 IVR 测试。

PJSIP2.1 的版本（支持 silk 编码）也可以依照上面的详细修改进行修改，本人测试一样可以使用。

118. FreeSwitch 在语音转码通话模式下 CPU 开销如何？

测试的目标是测试 FreeSwitch 在转码通话情况下的并发性能,测试使用的 FreeSwitch 版本是 1.2.10 使用 64 位模式 X64 进行编译。本次测试考虑到实际情况选择 iLBC 到 G711 或者 SILK 到 G711 两种转码情况，进行并发性能测试。

先准备 9 台测试客户端机器（因为跑 iLBC 或者 SILK 编码器，PJSIP 一台跑不了多少线比较差的机器通常只能跑 60 线左右）和一台 FreeSwitch 服务器，FreeSwitch 服务器的配置还是上面那个 DELL R720 E5-2640*2cpu 16G 内存:

准备 pjsip1.6 版本，支持 ilbc 编码， 作为主叫发起电话。

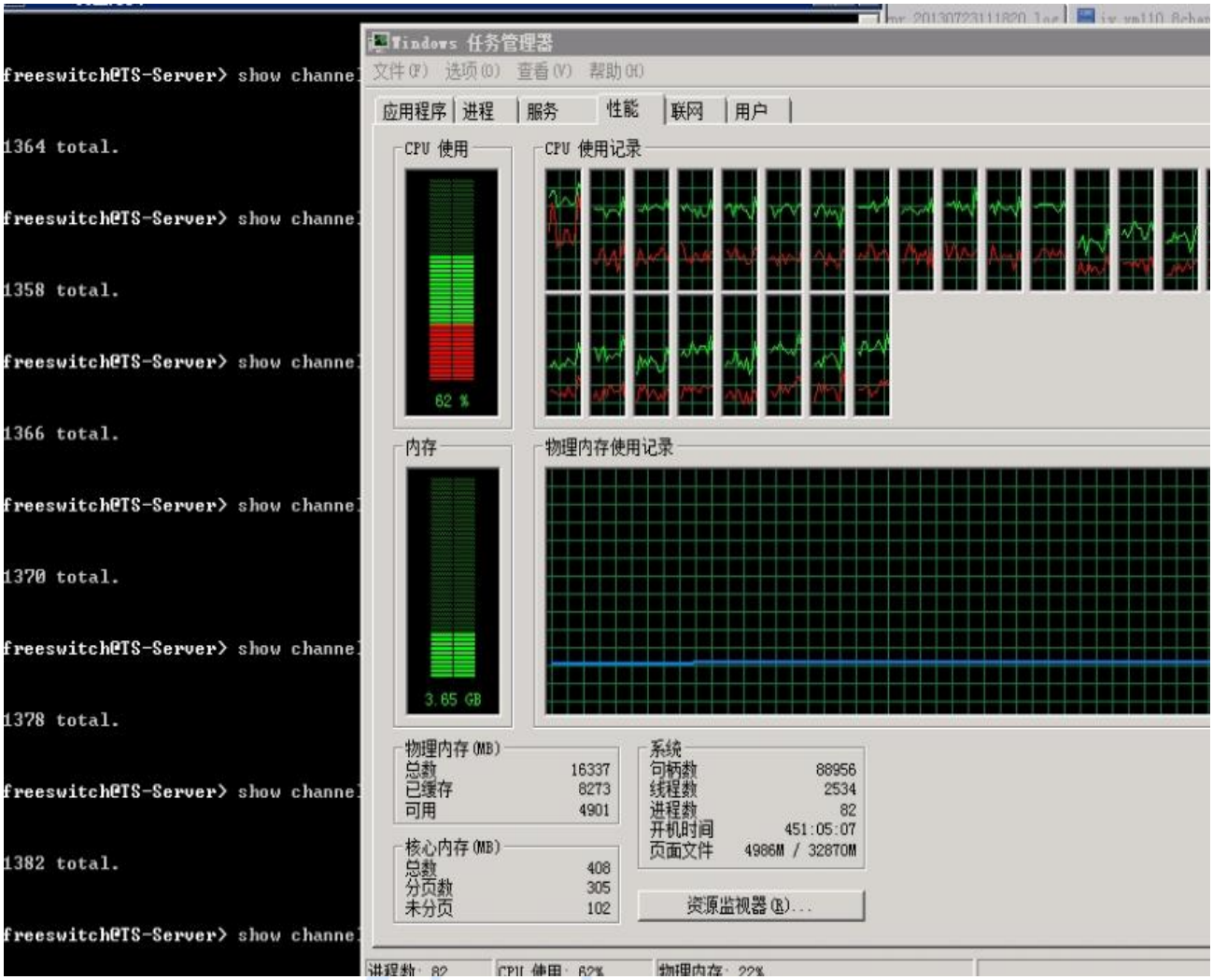
然后准备 sipp，使用 ulaw 编码，作为 FreeSwitch 转接出来的被叫电话。

测试场景是：使用各个客户端机器的 pjsip1.6 版本注册 FreeSwitch 到上，然后发起呼叫，呼叫接通接通之后 pjsip 播音，FreeSwitch 将来电转接到 sipp，sipp 作为被叫来电接通之后也是播放抓包文件，实际上也是播音。

然后 FreeSwitch 配置的编码器参数支持 iLBC 和 SILK 编码：

```
<X-PRE-PROCESS cmd="set" data="global_codec_prefs=PCMA,PCMU,GSM,SILK,iLBC"/>
  <X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMA,PCMU,GSM,SILK,iLBC"/>
```

首先测试 iLBC 到 G711 转码的情况下 CPU 的使用情况如下：



在并发 1382 的时候 CPU 大约 62%左右，运行一段时间，没有报错。

保守估计使用 iLBC 的转码通话这个配置两个 E5-2640CPU 的机器大约支持 1400 线并发。

接着测试 SILK 到 G711 转码的情况下，由于 pjsip1.6 不支持 SILK，因此使用 PJSIP2.1 的版本进行修改测试。修改方法同 pjsip1.6，不过设置 SILK 8000 的编码器优先级最高：上一个问题的代码里面

```
setcodec(1);
```

改为:setcodec(2);

另外额外编译 SILK 的 release 版本的 lib，然后编译 release 版本的 PJSIP。

然后使用 pjsip2.1 版本首选 silk 编码，作为主叫发起电话。

情况大体类似，SILK 到 ulaw 的转码看样子 CPU 开销要大一下，在 1284 线并发的使用 cpu 达到 66%，运行一段时间，开始出现下面错误：

```
freeswitch@TS-Server> show channels count
1284 total.
freeswitch@TS-Server> 2013-07-24 11:49:59.232800 [ERR] mod_silk.c:372 SKP_Silk_D
ecode returned -12!
2013-07-24 11:49:59.232800 [ERR] mod_silk.c:279  Silk Error: Payload has bit errors
2013-07-24 11:49:59.232800 [ERR] switch_core_io.c:1179 Codec SILK decoder error!
2013-07-24 11:49:59.242800 [ERR] mod_silk.c:372 SKP_Silk_Decode returned -12!
2013-07-24 11:49:59.242800 [ERR] mod_silk.c:279  Silk Error: Payload has bit errors
```

2013-07-24 11:49:59.242800 [ERR] switch_core_io.c:1179 Codec SILK decoder error!

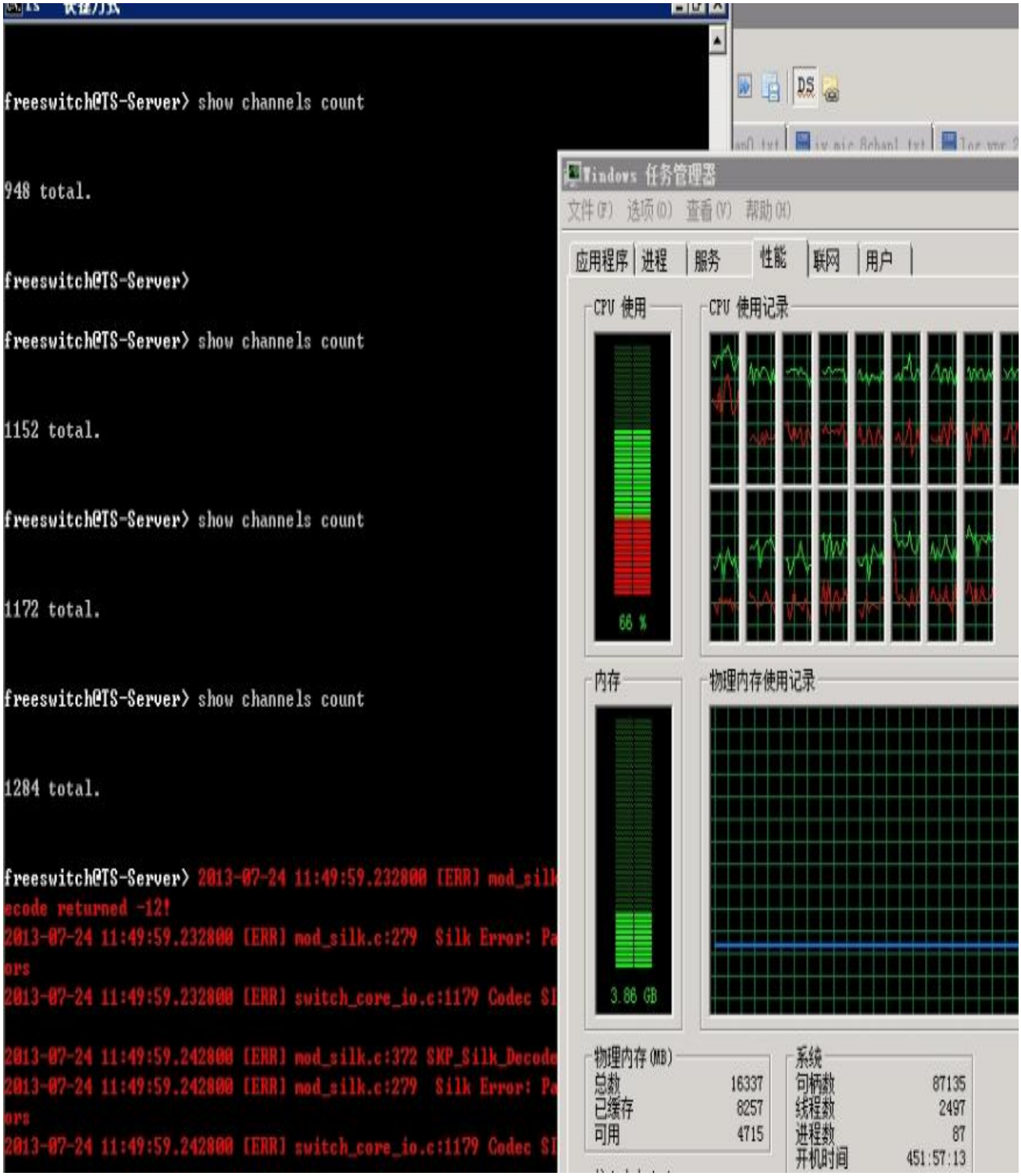
2013-07-24 11:49:59.242800 [ERR] mod_silk.c:372 SKP_Silk_Decode returned -12!

2013-07-24 11:49:59.242800 [ERR] mod_silk.c:279 Silk Error: Payload has bit errors

2013-07-24 11:49:59.242800 [ERR] switch_core_io.c:1179 Codec SILK decoder error!

可见使用 SILK 的转码情况，FreeSwitch 在同样的机器上支持的线路要少，而且会报解码错误（也可能是由于客户端 CPU 太忙导致，因为客户端上 PJSIP 也有看到错误提示）。

测试过程，FreeSwitch 的 CPU 的使用情况如下：



保守估计使用 SILK 的转码通话这个配置两个 E5-2640CPU 的机器大约支持 1000 线并发。

119. 如何在 32 位的 Windows 机器上让 FreeSwitch 支持超过 2G 的内存？

FreeSwitch 所有的稳定的版本从 1.2.1 到 1.2.14 在实际使用中在 32 位的 windows 机器上一个进程通常超过 1.6G 以上就会随时崩溃飞掉。
由于 FreeSwitch 在通话过程中存在内存一直增加（泄漏）的情况（一直到挂机了增加的内存才释放）。（此 BUG 本人搞了 3 天都未能找到是哪里泄漏。）
假如是坐席一直保持在线的使用模式（呼叫中心或者电话营销通常都是这种使用模式），内存就会一直上涨。实际使用环境中 150 个的坐席在线，内存就会很容易达到 1.5G 从而导致 FreeSwitch 随时可能崩溃。最终的方案当然是使用 64 位的操作系统，但是在某些不能更换 XP 和 2003 等 32 位操作系统的情况（很多人都知道 2008 的 D 版不是那么好搞的。）下怎么办？
通过下面这个方法，加上配合上面的“FreeSwitch 如何设置会话超时时间，如何定时挂机？”的办法让坐席上线 4-5 个小时之后就下线（中午或者晚上休息吃饭）。
组合成了一个凑合的解决办法基本可以对付。
以 Windows 2003 为例，要想办法使进程能有超过 2G 的内存需要下面两个步骤：
1. Windows 2003 或者 XP 操作系统，修改 c:\boot.ini 文件在/fastdetect 之类的后面加上 /3GB
如下：win2003

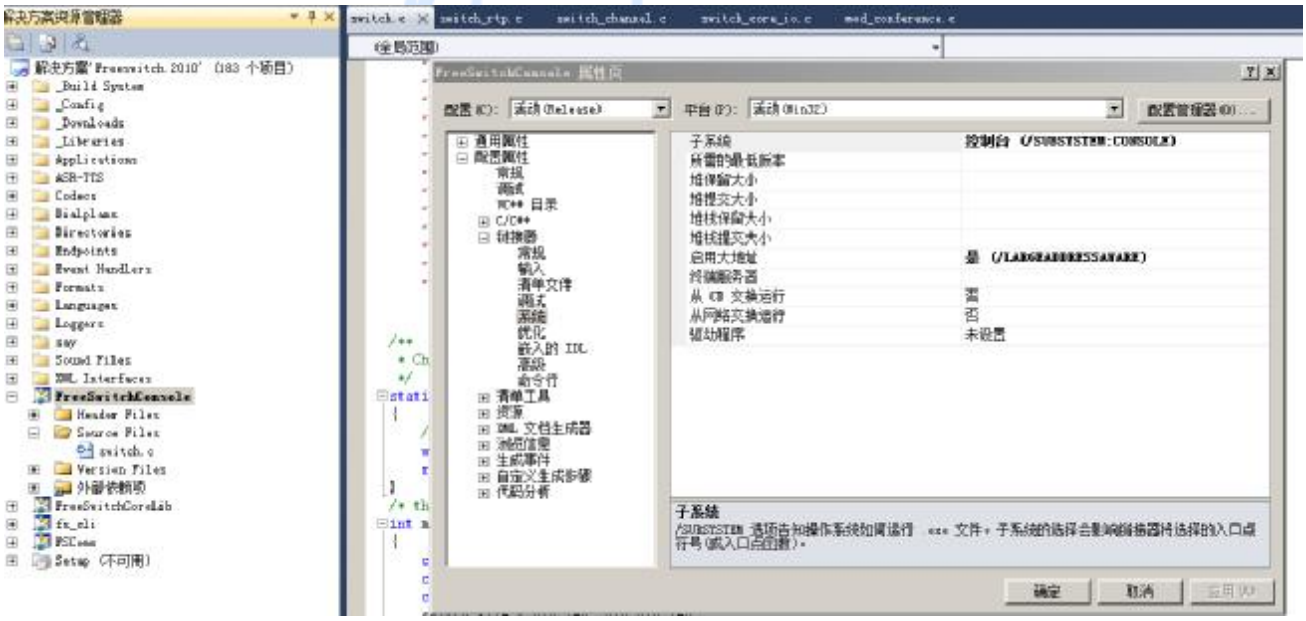
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise" /fastdetect /3GB

假如是 WIN7 操作系统：

在命令行下运行： bcdedit /set increaseuserva 3072 进行设置。

然后重启机器。

2.在 VS 编译环境里面，修改 FreeSwitch 的所有的工程文件，改为启用大地址 LargeAddressAware 支持，参照下图：记住是所有有输出 DLL 和 EXE 的工程都要设置。



实际测试，修改之后的 FreeSwitch，在 32 位的机器上可以支持到 2.2G-2.4G 左右。
再次强调，在此 bug 没有解决之前，最终的方案还是使用 64 位的 os 比较靠谱。

第六章 FreeSwitch 异常测试部分

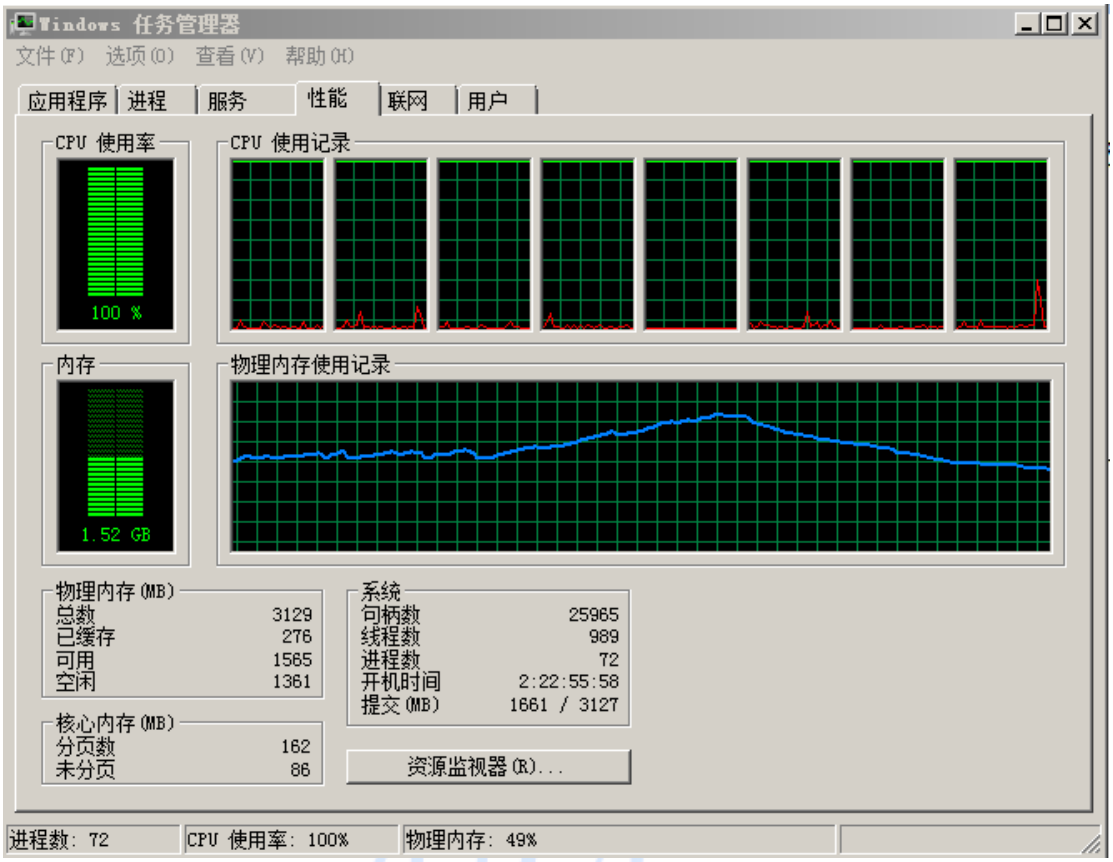
异常测试环境说明：
异常测试目的尽量模拟实网环境，使用 mysql 作为工作数据。运行 FreeSwitch 服务器的配置是 WIN7 32 位 3612 CPU，6G 内存，由于 32 位 OS 的限制。实际 OS 能使用的不到 4G，其它的 2G 额外的内存划做内存盘 Z：盘。
模拟实网运行过程中发生的各种异常，特别关注的是异常发生的时候，以及异常消失之后，FreeSwitch 能否正常运行。
测试使用的是 IVR 模式进行测试。ESL 的 IVR 程序和 FreeSwitch 在一个机器上运行。测试的场景是 拨入，播音，按键 1 播音 挂机，按键 2 转出，或者按键#进入电话会议。
呼叫测试的客户端使用 PJSIP。
开了 3 个 PJSIP 进程，分别模拟按键 1 播音 挂机，按键 2 转出，和按键#进入电话会议
并发在 100 线到 200 线之间。

120. FreeSwitch 使用过程中 cpu 负载发生异常结果如何？

先准备一个吃 CPU 的测试程序
测试机器是 8 核，因此，写了一个测试程序，
假如要吃 100%CPU 就开 8 个死循环线程。
假如要次 90%左右 CPU 就开了 7 个死循环的线程，仅仅留一个核给 FreeSwitch 使用：

```
int CallOutWorkThreadRealTime(void* arg)
 {
  int   i;
  while (1) {
        i=0;
        i++;
  }
}
int main(int argc,char *argv[])
{
  for(int i=0;i<7;i++)
CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)CallOutWorkThreadRealTime,(LPVOID)(0),0, NULL);
  while(1)
  {
        Sleep(10);
  }
  return 0;
}
```

Cpu 异常有两种情况：
第一种是 FreeSwitch 正常运行过程中，使用其它吃 CPU 的测试进程把 100%cpu 占用了。如下图：

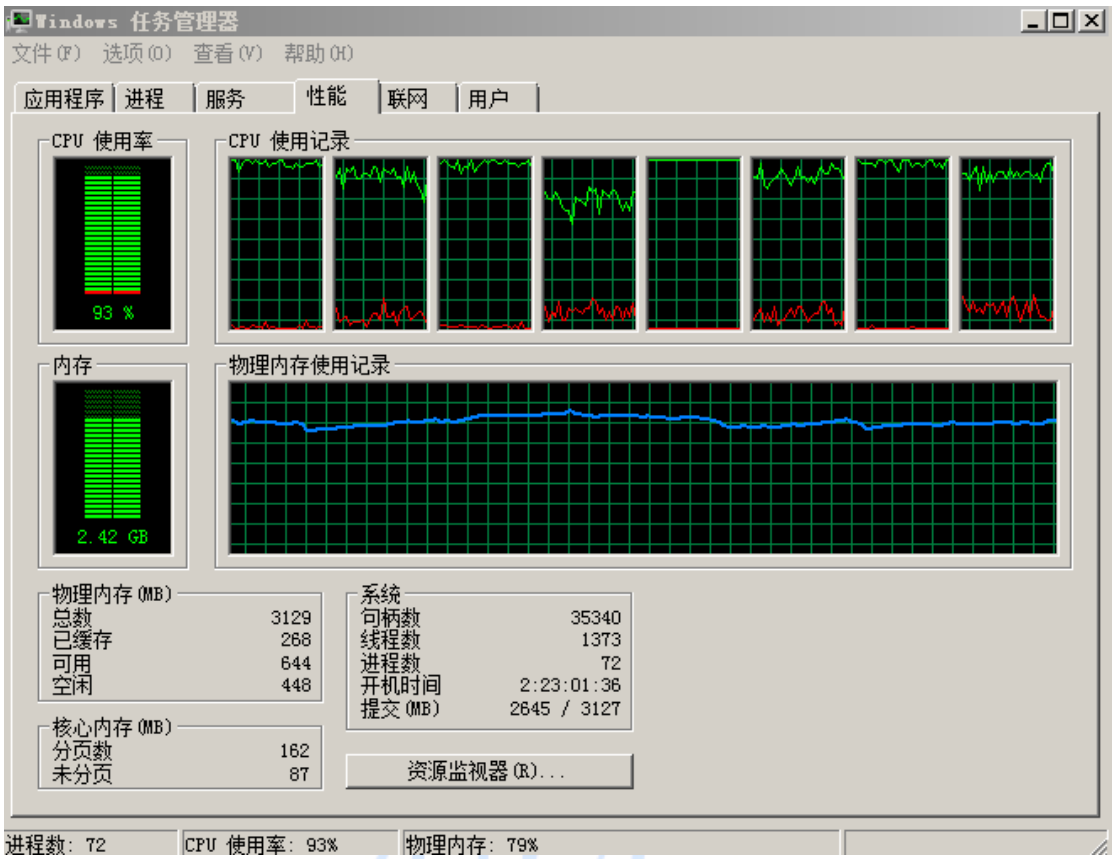


发生这种情况，FreeSwitch 不再接受任何拨入的呼叫。尝试使用额外的 eyebeam 测试无法拨入。通道并发数目一直下降：

```
freeswitch@yhy-PC32> show channels count
81 total.
freeswitch@yhy-PC32> show channels count
79 total.
freeswitch@yhy-PC32> show channels count
50 total.
freeswitch@yhy-PC32> show channels count
0 total.
```

其它吃 CPU 的测试进程关闭之后，FreeSwitch 马上可以接受拨入。

第二种是 FreeSwitch 正常运行过程中，使用其它吃 CPU 的测试进程把 90%cpu 占用了。



这种情况下 FreeSwitch 可以接受呼叫，但是进程有告警提示：

freeswitch@yhy-PC32> 2013-07-21 21:26:10.578128 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-07-21 21:26:10.598129 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-07-21 21:26:10.618131 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-07-21 21:26:10.658133 [CRIT] mod_sofia.c:4831 Error Creating Session

2013-07-21 21:26:10.698135 [CRIT] mod_sofia.c:4831 Error Creating Session

尝试修改 conf\autoload_configs\switch.conf.xml 配置里面的 CPU 控制参数：

<param name="min-idle-cpu" value="1"/>

或者

<param name="min-idle-cpu" value="0"/>

都无效。

其它吃 CPU 的测试进程关闭之后，FreeSwitch 不在出现错误告警。

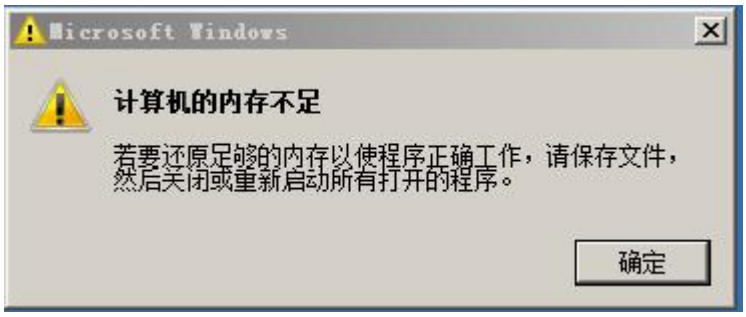
121. FreeSwitch 使用过程中内存负载发生异常结果如何？

先准备一个吃内存的测试程序，在 FreeSwitch 正常运行过程中启动。

```
int main(int argc,char *argv[])
{
 while(1)
 {
       char *ch= new char[102400];
       Sleep(10);
 }
```

```
 return 0;
}
```

跑起来一段时间之后 OS 出现提示：



2013-07-21 20:56:59.880122 [ERR] mod_sndfile.c:202 Error Opening File [d:/data/system/bye.alaw] [Internal malloc () failed.]

2013-07-21 20:56:59.900123 [ERR] mod_sndfile.c:202 Error Opening File [d:/data/system/bye.alaw] [Internal malloc () failed.]

2013-07-21 20:56:59.940125 [CRIT] mod_conference.c:7207 Memory Error Creating Audio Buffer!

2013-07-21 20:56:59.940125 [ERR] mod_sndfile.c:202 Error Opening File [d:/data/system/bye.alaw] [Internal malloc () failed.]

2013-07-21 20:58:36.827216 [WARNING] sofia.c:1639 MSG Thread 0 Started

2013-07-21 20:58:40.469424 [ERR] mod_lua.cpp:198 Z:/Release/scripts/gen_dir_user_xml.lua:47: attempt to concatenate local 'req_user' (a nil value)

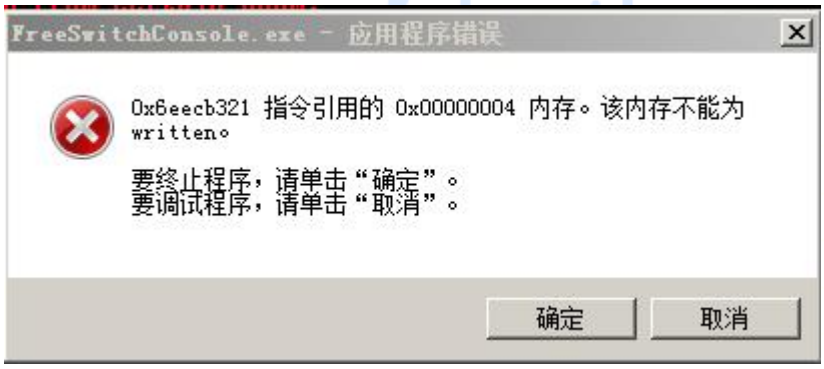
stack traceback:

Z:/Release/scripts/gen_dir_user_xml.lua:47: in main chunk

2013-07-21 20:58:40.469424 [ERR] mod_lua.cpp:264 LUA script parse/execute error!

。。。

然后 FreeSwitch 崩溃



重新启动 FreeSwitch ，重复进行上面的测试。

情况大体类似，日志稍微不同，最终的结果还是崩溃：

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1698 Thread Failure!,res=721455

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1657 LUKE: I'm hit, but not bad.

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1658 LUKE'S VOICE: Artoo, see what you can do with it. Hang on back there....

Green laserfire moves past the beeping little robot as his head turns. After a few beeps and a twist of his mechanical arm,

Artoo reduces the max sessions to 145 thus, saving the switch from certain doom.

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1698 Thread Failure!,res=721455

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1657 LUKE: I'm hit, but not bad.

2013-07-21 21:01:40.905745 [CRIT] switch_core_session.c:1658 LUKE'S VOICE: Artoo, see what you can do with it. Hang on back there....

Green laserfire moves past the beeping little robot as his head turns. After a few beeps and a twist of his mechanical arm,

Artoo reduces the max sessions to 145 thus, saving the switch from certain doom.

2013-07-21 21:02:04.046068 [ERR] switch_odbc.c:365 STATE: HY000 CODE 2005 ERROR: [MySQL][ODBC 3.51 Driver]Unknown MySQL server host 'localhost' (11004)

2013-07-21 21:02:04.046068 [CRIT] switch_odbc.c:280 The sql server is not responding for DSN mysql [][230]

2013-07-21 21:02:04.046068 [CRIT] switch_odbc.c:288 The connection could not be re-established

2013-07-21 21:02:04.166075 [WARNING] switch_core_state_machine.c:514 60c61b86-db38-4391-9c42-d0d3613220cc sofia/internal/1001@192.168.1.102 Abandoned

2013-07-21 21:02:05.046125 [CRIT] switch_odbc.c:280 The sql server is not responding for DSN mysql [STATE: 08003 CODE 0 ERROR: [Microsoft][ODBC 驱动程序管理器] 连接未打开

][230]

2013-07-21 21:02:08.585328 [CRIT] switch_core_session.c:1698 Thread Failure!,res=721455

2013-07-21 21:02:08.585328 [CRIT] switch_core_session.c:1657 LUKE: I'm hit, but not bad.

2013-07-21 21:02:08.585328 [CRIT] switch_core_session.c:1658 LUKE'S VOICE: Artoo, see what you can do with it. Hang on back there....

Green laserfire moves past the beeping little robot as his head turns. After a few beeps and a twist of his mechanical arm,

Artoo reduces the max sessions to 132 thus, saving the switch from certain doom.

这个内存异常说明 FreeSwitch 严重依赖内存。

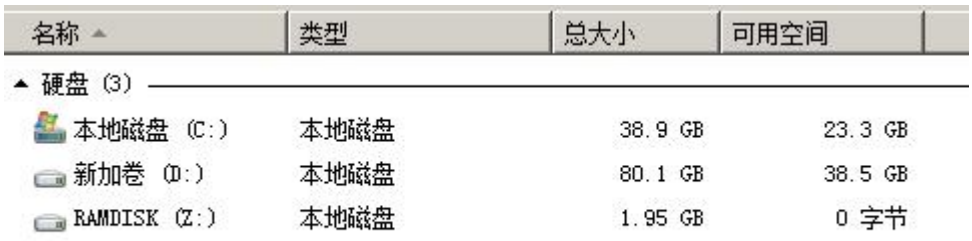
对内存控制做的不够完善。

实网运营的时候要十分注意机器的内存使用情况。一旦有其它进程发生内存泄漏就可能导致 FreeSwitch 崩溃。解决办法之一就是 FreeSwitch 单独一个物理机器运行。

122. FreeSwitch 使用过程中硬盘容量发生异常结果如何？

把 FreeSwitch 跑在 Z：盘上，OS 的 temp 临时目录也是放在 Z:

通过参数关闭日志输出，然后 FreeSwitch 运行一段时间之后把硬盘都占用到没有任何硬盘空间：



名称	类型	总大小	可用空间
▲ 硬盘 (3)			
本地磁盘 (C:)	本地磁盘	38.9 GB	23.3 GB
新加卷 (D:)	本地磁盘	80.1 GB	38.5 GB
RAMDISK (Z:)	本地磁盘	1.95 GB	0 字节

硬盘没有容量，测试 FreeSwitch 可以正常运行。

123. FreeSwitch 使用过程中服务器的机器网络发生异常结果如何？

FreeSwitch 网络断开：FreeSwitch 正常，没有崩溃。
网络恢复之后 FreeSwitch 正常运行，可以继续使用。
但是可能存在通道吊死的情况，比如：
会场里面的通道就一直在会场，不会挂机
freeswitch@yhy-PC32> show channels count
37 total.
因此应用上要做超时机制。否则可能导致话路吊死在里面 造成计费产生超长话单。
吊死的通道在重启业务流程之后，清除挂机。

124. FreeSwitch 使用过程中客户端的网络发生异常结果如何？

3*60 线的 pjsip=180 路
客户端网络异常，网线拔出，或者网络禁用
一段时间之后插上，FreeSwitch 正常。
但是可能存在通道吊死的情况。

125. FreeSwitch 使用过程中客户端的发生异常结果如何？

3*60 使用使用任务管理器 kill 客户端，或者直接关闭测试客户端
FreeSwitch 正常。
但是可能存在通道吊死的情况。

126. FreeSwitch 使用过程中异常客户端发生攻击结果如何？

第一种攻击是：乱发注册的消息，瞎蒙帐号密码的情况。

测试 FreeSwitch 正常。

第二种攻击是：冒充的 ip 错误情况。

ImsDroid 的 里面的 public Identiry 里面的 sip:1031@192.168.1.102

写成 sip:1031@1192.168.1.102

或者

写成 sip:1031@192.1168.1.102

或者

写成 sip:1031@1192.168.1111.102

其它的配置都是对的情况下，点签入，FreeSwitch 会一直循环输出：

2013-07-21 22:34:05.838051 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request from 192.168.1.103:39876

2013-07-21 22:34:05.838051 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request from 192.168.1.103:39876

2013-07-21 22:34:05.838051 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request from 192.168.1.103:39876

无穷无尽。

直接导致 FreeSwitch 服务受到影响，虽然没有停止服务，但是响应速度等大大降低，严重影响了系统的使用。

更加要命的是就算用户的密码是错误的，IP 地址的前面 3 段随便一段写超过了

也会发生上面的情况。这样用户只要知道 FreeSwitch 的 IP 和端口就可以随便进行攻击。

第七章 FreeSwitch FlashPhone 部分

127. 什么是 flash phone/SIP?

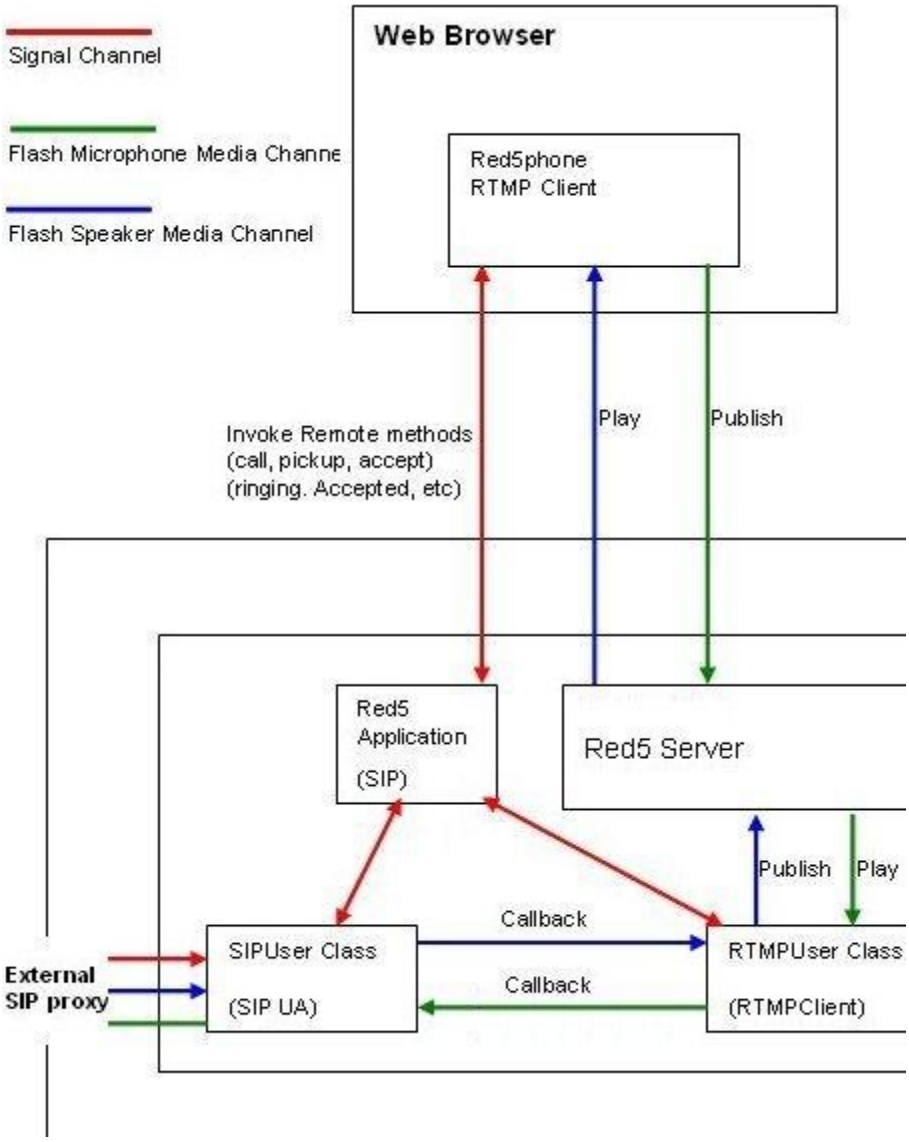
Flash phone 就是通过网页上的 flash 插件来实现语音媒体通话的功能。作为搞 voip sip 的研发人员，了解这个有一定的必要性。

在 webrtc 没有完善之前，这个是一个浏览器上跑 VOIP 的替代方案。只要网络带宽有保证，跑起来还是很溜的。我认为 支持 rtmp 协议能进行语音通话的系统都可以叫做 flash phone。

其代表是开源的 red5phone ，它使用 java 基于 red5 媒体服务器上开发的 实现 flash 到 sip 转换网关。网址是：

http://code.google.com/p/red5phone/

red5phone 基本系统结构图如下：



从图中可以大概了解 FLASH PHONE 的服务端的工作原理

128. 为啥需要 flash phone/SIP？

Flash sip 或者 flash phone 的优势是在浏览器里面使用 flash player 插件
不需要新安装插件。这对用户是非常方便的。

129. 啥情况下下需要 flash phone？

出现的用途有几种场景：

A. 访问系统的用户是非特定的用户或者游客。它们通过网页上对系统进行访问。使用 activeX 插件的形式，用户会不信任，也不会安装。

因此可以通过 Flash sip 或者 flash phone 来对系统进行语音访问。

因为绝大多数的机器浏览器都安装了 Flash player 插件。

通俗一点描述：用户浏览某个网站，对里面的某个产品有兴趣，可以点产品边上的电话按键马上和客服或者销售人员进行语音沟通。

以前类似的业务是通过 web 800 或者 web call 让用户输入手机号码然后系统呼叫用户的手机跟坐席进行通话来实现的。许多用户不想泄漏手机，不会输入手机号码。

B. B/S 结构的呼叫中心，系统的客服或者坐席是经常变换机器的，它们今天可能在公司上班，明天可能在家里上班。

C. 有些地方的运营商不允许使用 SIP 进行 VOIP 通讯。比如山东聊城 联通的宽带就是禁止 使用 SIP 进行 VOIP 通讯。这个时候通过 Flash phone 作为媒体通讯是合适的。

130. 使用 flash phone 使用什么协议？

flash phone 使用 adobe 公司的 flash player
插件，跟系统使用 RTMP 进行通信
RTMP 是基于 TCP 协议开发的一种媒体协议。
客户端就是 flash player 或者基于 AIR 的应用程序，服务器可以说 FMS，RED5, 和 FreeSwitch 的 MOD_RTMP

131. 使用 flash phone 需要注意哪些问题？

由于 RTMP 是基于 TCP 协议开发的一种媒体协议，
TCP 的协议 网络影响特别大。
因此使用 flash phone 的时候需要注意：

A. 系统的服务端和客户端最好能在一个运营商内使用。跨运营商带宽是不能保证的，延时也会明显加大。在使用 tcp 进行媒体传输通讯的时候影响会显著放大。

B. 假如要公众服务的系统，最好是架设在 BGP 的四线接入的机房内部，才能有效保证 接入的带

宽和延时。

C. RTMP 默认是使用 1935 端口。机器上要保证这个端口是没有进程其它占用的。

D. flash phone 测试呼叫 9195　延迟 5 秒的回音测试，是不行的，有 BUG。

132. FreeSwitch 如何增加 RTMP 接口协议模块以实现对 flash phone 的支持？

有些版本的 FreeSwitch 默认是不提供 mod_rtmp 模块，假如没有需要自己在工程文件里面加上，然后自己编译。

编译好 mod_rtmp 模块

之后通过修改 \conf\autoload_configs

目录下的 modules.conf.xml 配置文件：

```
<!-- <load module="mod_rtmp"/>  -->
```

改为：

```
<load module="mod_rtmp"/>
```

然后保存。重启 FreeSwitch 就可以使用。

133. FreeSwitch 的 flash 配置文件是哪个，如何配置 RTMP 的端口？

\conf\autoload_configs 目录下的

rtmp.conf.xml 就是这模块的配置文件。

配置文件里面的：

```
<param name="bind-address" value="0.0.0.0:1935" />
```

其中的 1935 就是 rtmp 模块发默认端口。假如 1935 端口被占用，

就要修改这个参数。

134. FreeSwitch 的 rtmp 如何配置不需要 login 就可以呼叫其它分机？

首先：

修改 \conf\autoload_configs 目录下的 rtmp.conf.xml 配置文件。

配置文件里面的：

```
<param name="auth-calls" value="true" />
```

参数 改为：

```
<param name="auth-calls" value="false" />
```

然后不用 flash phone 上 login 就可以呼叫。

但是这样是有风险的，我们建议除非是内网使用，否则不要设置为 false。

然后在 flex 代码里面 在 makeCall 呼叫的时候 account 设置为”,flex 代码 makeCall2 函数如下：

```
public function makeCall2(number:String, account:String):int {
                if (netConnection != null)
                {
                    hangup(now_uuid);
                    if (incomingNetStream == null)
                    {
                        setupStreams();
                    }
                    if(loginok)
                        netConnection.call("makeCall", null, number, account);
                    else
                        netConnection.call("makeCall", null, number, '');
                    return 0;
                }
                return -1;
        }
```

注意上面的斜体部分：

```
if(loginok)
                netConnection.call("makeCall", null, number, account);
 else
                netConnection.call("makeCall", null, number, '');
```

假如登录过，那么使用 account 帐号作为主叫进行呼叫，假如没有登录过，那么使用”帐号作为主叫进行呼叫，这个时候被叫看到的主叫号码是 默认的主叫号码:’0000000000’.没有登录假如使用 netConnection.call("makeCall", null, number, account);进行呼叫是呼叫不通的。

135. FreeSwitch 的 flash phone 使用啥工具进行修改开发？

提供的 支持 rtmp 的 代码在 \clients\flex 目录下
flash phone 是基于 FLEX 开发的目前支持
Adobe Flash Builder 4.6 开发。

136. FreeSwitch 的 flash phone 代码哪些是最有用的？

FreeSwitch 默认提供的代码很庞大，功能很多，尤其是 JS 部分，给初学者带来很大的困惑。实际上基本的通话功能只需要很少的 code，
简化之后的 code 只有 300 多行。
整个 demo 可以到根据附录去下载。

137. FreeSwitch 的 flash phone 如何呼叫分机？

flash phone 连接 connect 到系统之后，可以呼叫已经注册到

FreeSwitch 上的分机。
呼叫的 方法是 直接呼叫：
sip:分机@FreeSwitch 的 IP

138. FreeSwitch 的 flash phone 如何查哪些机器连接上来？

在 FreeSwitch cli 界面上输入：
FreeSwitch>rtmp status profile default sessions
假如都没有用户连接 connect 上来，提示如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0
Sessions:
uuid,address,user,domain,flashVer
假如用户已经 connect 上来,但是没有 login，提示类似如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0
Sessions:
uuid,address,user,domain,flashVer
3e0d3f92-0980-427f-8c49-786f5c6c6f62,192.168.1.102:16671,,,WIN 11,4,402,287

假如用户已经 connect 上来，并且已经 login 提示类似如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0
Sessions:
uuid,address,user,domain,flashVer
3e0d3f92-0980-427f-8c49-786f5c6c6f62,192.168.1.102:16671,1001,192.168.1.102,WIN 11,4,402,287
注意里面的 IP 地址:PORT 是客户端的 ip 和端口。

139. FreeSwitch 的 flash phone 如何查注册上来的分机？

在 FreeSwitch cli 界面上输入：FreeSwitch> rtmp status profile default reg
假如没有用户注册上来（login）提示如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0
假如有用户注册上来提示如下：

Profile: default

I/O Backend: tcp

Bind address: 0.0.0.0:1935

Active calls: 0

Registrations:

user,nickname,uuid

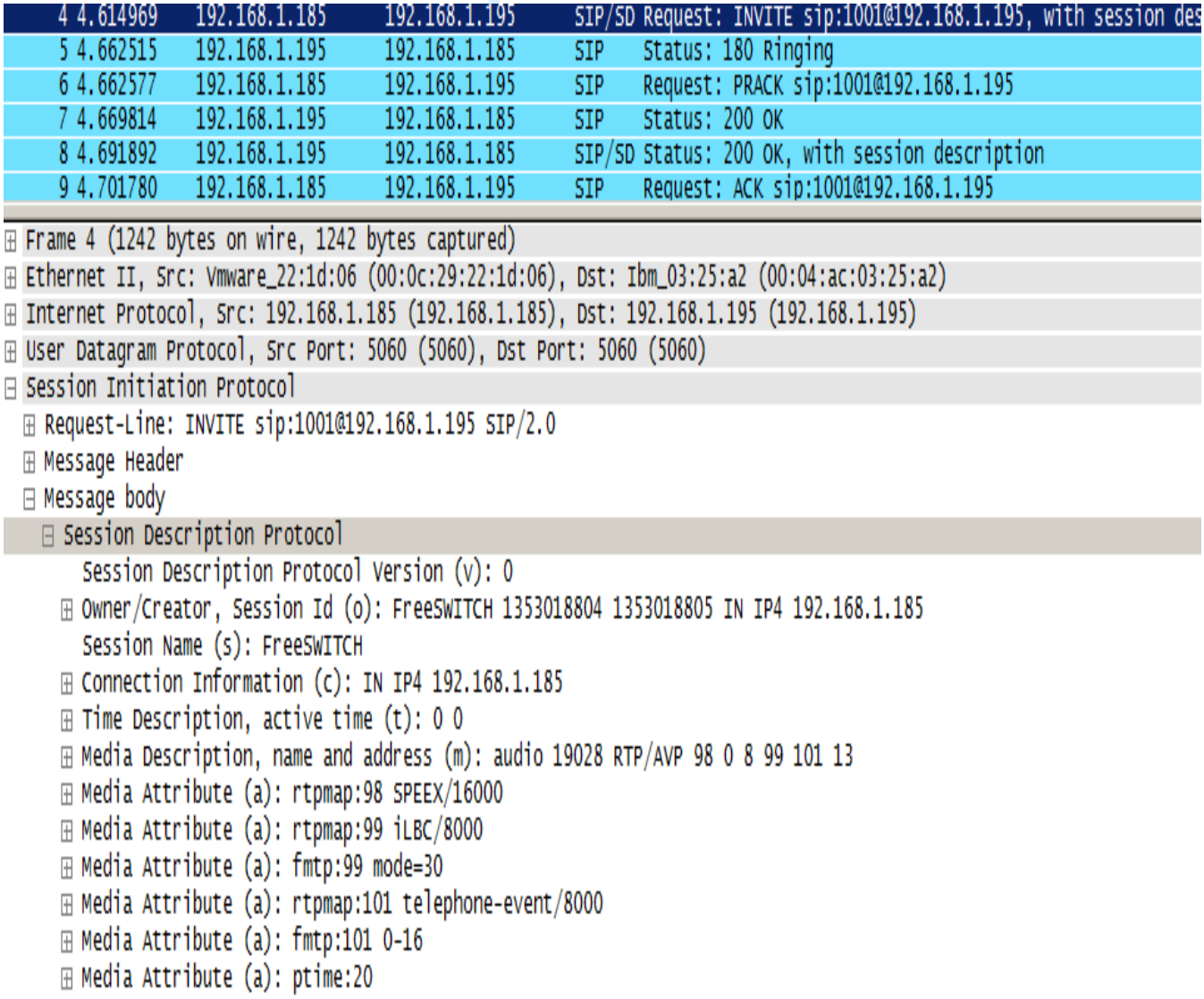
1002@192.168.1.102,1002@192.168.1.102,6873e7ae-572b-4166-871c-54bc3189acb7

1001@192.168.1.102,1001@192.168.1.102,3e0d3f92-0980-427f-8c49-786f5c6c6f62

其中 1002 和 1001 就是注册上来的分机

140. FreeSwitch 的啥情况下 iLBC 编码不能使用？

本人测试的 1.2.1，1.2.3 1.2.5.3 的版本，由于 FreeSwitch 的 bug 导致某些情况下（如 flash phone 客户端）通过 iLBC 编码呼叫其它系统，或者其它软电话分机的时候，INVITE 里面的 SDP 是错误的。如下图：



实际上 iLBC 编码 的 RTP MAP 应该是 98.在这里 flash phone 默认的 SPEEX 的 RTP MAP 变成了 98.而 iLBC 编码 的 RTP MAP 变成了 99.从而导致无法跟其它系统（比如毅航的 ISX 系列）集成使用，比如会造成单通。这个时候需要通过修改源码，重新编译 mod_sofia 模块 替换之后 才能使用。

具体修改的地方是 sofia_glue.c 文件里面的sofia_glue_set_local_sdp函数里面的
下面部分：

```
if (!tech_pvt->payload_space) {
        int i;
        tech_pvt->payload_space = 98;
```

Ver 1.2.3:

mod_rtmp.c

```
#define SPEEX_BAND_YHY    8000
…
switch_core_timer_init(&tech_pvt->timer, "soft", 20, (SPEEX_BAND_YHY / (1000 / 20)),
switch_core_session_get_pool(session));

    /* Initialize read & write codecs */
    if (switch_core_codec_init(&tech_pvt->read_codec, /* name */ "SPEEX",
          /* fmtp */ NULL,    /* rate */ SPEEX_BAND_YHY, /* ms */ 20, /* channels */ 1,
        /* flags */ SWITCH_CODEC_FLAG_ENCODE | SWITCH_CODEC_FLAG_DECODE,
        /* codec settings */ NULL, switch_core_session_get_pool(session)) != SWITCH_STATUS_SUCCESS) {
        switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "Can't initialize read codec\n");

        return SWITCH_STATUS_FALSE;
    }

    if (switch_core_codec_init(&tech_pvt->write_codec, /* name */ "SPEEX",
          /* fmtp */ NULL,    /* rate */ SPEEX_BAND_YHY, /* ms */ 20, /* channels */ 1,
        /* flags */ SWITCH_CODEC_FLAG_ENCODE | SWITCH_CODEC_FLAG_DECODE,
        /* codec settings */ NULL, switch_core_session_get_pool(session)) != SWITCH_STATUS_SUCCESS) {
        switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "Can't initialize write codec\n");

        return SWITCH_STATUS_FALSE;
    }
    ….
```

在修改代码之后 测试

iLBC 和 speex@8000h@20i 不能同时设置到 编码器里面，否则:

eyebeam(软电话的)呼叫会导致 ilbc 的 rtpmat=97 ， speex 的 rtpmap=98

flashphone 的呼叫会导致 speex 的 rtpmap=97，ilbc 的 rtpmat=98 导致不一致的错误

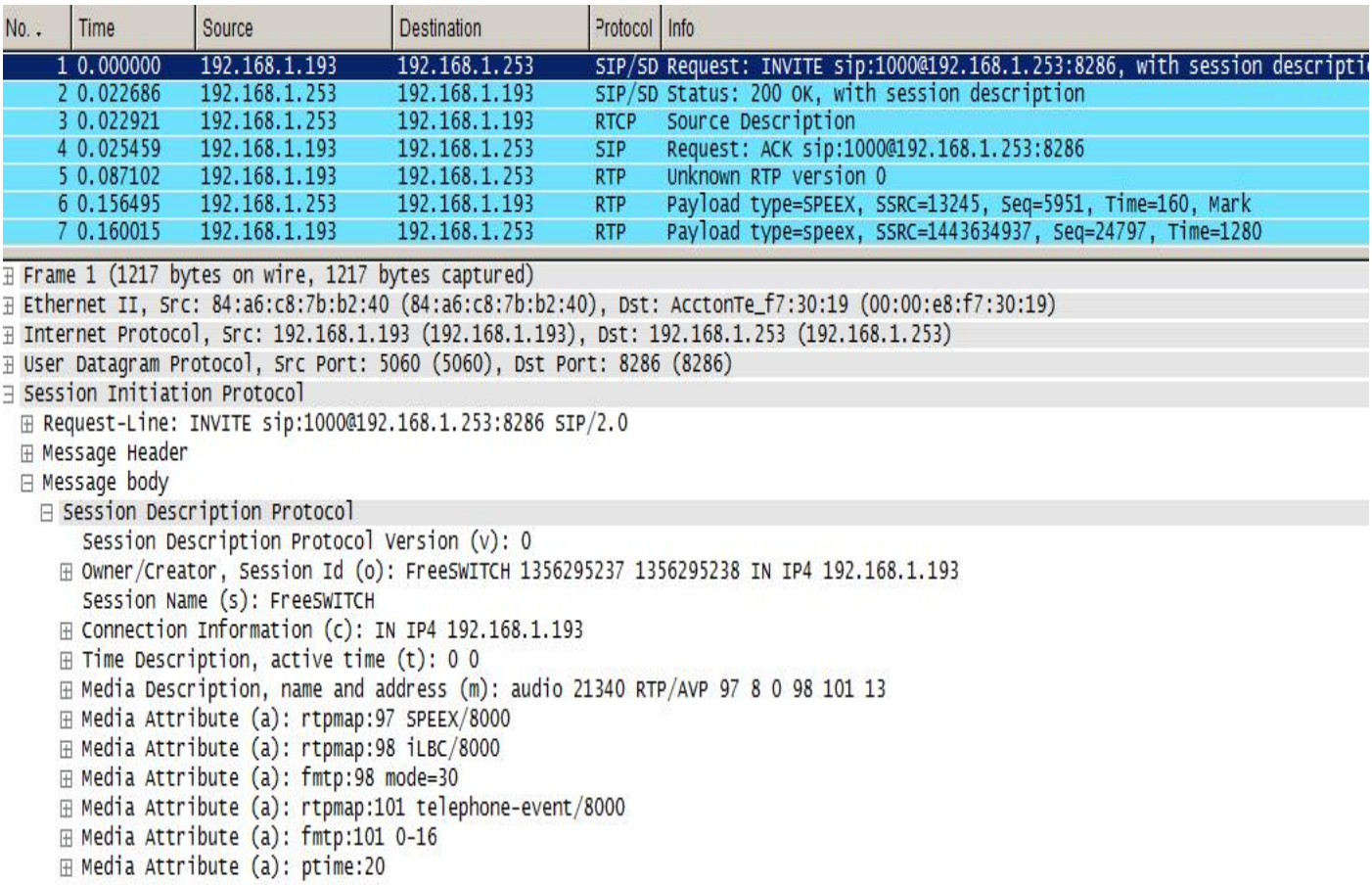
因此建议只能 使用 iLBC 或者 SPEEX 里面的一个

假如没有设置 speex@8000h@20i ，flash 呼叫内部分机的时候 SDP 不会出现:

speex 的 rtpmap=97

但是假如是 flash 呼叫 外部的没有注册上来的 sip，那么会出现 speex 的 rtpmap=97 的 sdp 如下图：

呼叫 sip:1000@192.168.1.253:8286



No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.193	192.168.1.253	SIP/SD	Request: INVITE sip:1000@192.168.1.253:8286, with session descripti
2	0.022686	192.168.1.253	192.168.1.193	SIP/SD	Status: 200 OK, with session description
3	0.022921	192.168.1.253	192.168.1.193	RTCP	Source Description
4	0.025459	192.168.1.193	192.168.1.253	SIP	Request: ACK sip:1000@192.168.1.253:8286
5	0.087102	192.168.1.193	192.168.1.253	RTP	Unknown RTP version 0
6	0.156495	192.168.1.253	192.168.1.193	RTP	Payload type=SPEEX, SSRC=13245, Seq=5951, Time=160, Mark
7	0.160015	192.168.1.193	192.168.1.253	RTP	Payload type=speex, SSRC=1443634937, Seq=24797, Time=1280

```
⊞ Frame 1 (1217 bytes on wire, 1217 bytes captured)
⊞ Ethernet II, Src: 84:a6:c8:7b:b2:40 (84:a6:c8:7b:b2:40), Dst: AcctonTe_f7:30:19 (00:00:e8:f7:30:19)
⊞ Internet Protocol, Src: 192.168.1.193 (192.168.1.193), Dst: 192.168.1.253 (192.168.1.253)
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 8286 (8286)
⊟ Session Initiation Protocol
  ⊞ Request-Line: INVITE sip:1000@192.168.1.253:8286 SIP/2.0
  ⊞ Message Header
  ⊟ Message body
    ⊟ Session Description Protocol
        Session Description Protocol Version (v): 0
      ⊞ Owner/Creator, Session Id (o): FreeSWITCH 1356295237 1356295238 IN IP4 192.168.1.193
        Session Name (s): FreeSWITCH
      ⊞ Connection Information (c): IN IP4 192.168.1.193
      ⊞ Time Description, active time (t): 0 0
      ⊞ Media Description, name and address (m): audio 21340 RTP/AVP 97 8 0 98 101 13
      ⊞ Media Attribute (a): rtpmap:97 SPEEX/8000
      ⊞ Media Attribute (a): rtpmap:98 iLBC/8000
      ⊞ Media Attribute (a): fmtp:98 mode=30
      ⊞ Media Attribute (a): rtpmap:101 telephone-event/8000
      ⊞ Media Attribute (a): fmtp:101 0-16
      ⊞ Media Attribute (a): ptime:20
```

141. 软电话分机如何实现对 FreeSwitch 的 flash phone 的呼叫？

flash phone 连接到 FreeSwitch 的 MOD_RTMP 之后。

需要 login 才能被呼叫。

软电话分机要呼叫，需要修改拨号计划：

在 conf\dialplan 目录下的 default.xml 文件里面修改：

在

```
<!-- http://wiki.FreeSwitch.org/wiki/Dialplan_XML -->
<include>
  <context name="default">
```

这之后

增加：

```
    <extension name="Local_Extension2">
        <condition field="destination_number" expression="^(10[01][0-9])$">
<action application="export" data="dialed_extension=$1"/>
<action application="set" data="call_timeout=10"/>
<action application="set" data="hangup_after_bridge=true"/>
```

```
<action application="set" data="continue_on_fail=true"/>
<action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
<action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
        </condition>
      </extension>
```

注意两条 bridge 命令，因为测试发现 nod_rtmp 有 bug。导致呼叫到 flash 客户端有时候不成功。continue_on_fail=true 的情况下万一 bridge 不行，我们就再来一次 bridge,通常就可以搞定！
你要是不放心甚至可以考虑再来一次。:-）

142. 如何实现同时支持呼叫分机和 flash client 分机?

类似上面：

```
  <include>
    <context name="default">
    <extension name="Local_Extension2">
         <condition field="destination_number" expression="^(10[01][0-9])$">
     <action application="export" data="dialed_extension=$1"/>
     <action application="set" data="call_timeout=10"/>
     <action application="set" data="hangup_after_bridge=true"/>
     <action application="set" data="continue_on_fail=true"/>
     <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
     <action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
     <action application="bridge" data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
         </condition>
       </extension>
```

注意有 3 条 bridge 命令，第一次

```
 <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
```

是先呼叫软电话的分机
要是没有软电话分机注册上，就呼叫 flash client 的分机。

第八章 FreeSwitch IM 消息聊天部分

143. FreeSwitch 是如何实现 IM 消息聊天功能？

FreeSwitch 的聊天功能是通过 SIP 协议的扩展来实现的，标准的 sip 协议不包括这个功能
根据 rfc3428 文档：
Session Initiation Protocol (SIP) Extension for Instant Messaging
FreeSwitch 的聊天是通过 sip 的 MESSAGE 消息来通信的。
比如 A 和 B 注册在 FreeSwitch 上，A 给 B 发消息，实际的过程如下：
A à SIP MESSAGE à FreeSwitch
A ß 200 OK ß FreeSwitch
FreeSwitch à SIP MESSAGE à B
FreeSwitch ß 200 OK ß B
默认情况下 FreeSwitch 开着就支持 终端到终端的聊天消息转发功能。

144. 哪些软电话支持 IM 聊天功能？

本人测试过以下软电话支持 IM 聊天功能：
Eyebeam
Linphone for windows
Linphone for IOS
Linphone for Android
CSipSimple for Android
以上这些软电话通过 FreeSwitch 互相发送 IM 消息假如对方都在线的话，是可以互相收到消息的。现在问题来了，假如有人不在线，默认情况下，FreeSwitch 丢弃消息。因此实用性大打折扣。咋办呢？下一个问题来回答这个问题。

145. 如何让 FreeSwitch IM 聊天支持离线消息功能？

这个问题比较复杂，基本上需要以下几个步骤：

1）修改配置启用 mod_sms 模块 conf\autoload_configs 目录下
modules.conf.xml 文件里面增加 下面内容：
<load module="mod_sms"/>

2）修改 conf\chatplan 目录下

default.xml 文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<include>
  <context name="default">
    <extension name="demo">
      <condition field="to" expression="^(.*)$">
        <action application="set" data="to=$1"/>
          <action application="send" />
      </condition>
    </extension>
  </context>
</include>
```

3）大动作来了：需要修改 FreeSwitch 源码

核心思想：拦截 对方不在线的时候的 IM 消息。通过 UDP 发送消息给其它程序模块（这里叫做 IM-SERVER）保存到数据库，下次对方注册上线的时候，通过 UDP 发送消息 IM-SERVER, IM-SERVER 通过查询数据库，找出之前的离线消息，发送给用户。启用了 mod_sms 之后程序才能发消息给用户。为啥使用 UDP，是因为：1 来不想影响到 FreeSwitch 的性能，假如直接写数据或者使用 TCP 可能会影响性能。2 来 IM-Server 跟 FreeSwitch 在一个机器上，UDP 丢包的概率很小。当然你也可以采用其它方案。

用户 A 给用户 B

发送离线消息的场景描述如下：

用户 A à FreeSwitch à UDP à IM-SERVER à DB

当用户 B 注册上线的时候 注册消息：FreeSwitch à UDP à IM-SERVER à DB

然后 IM-SERVER à ESL à FreeSwitch à 用户 B

首先发送离线消息修改

sofia_presence.c 文件：

修改如下函数：

```
switch_status_t sofia_presence_chat_send(switch_event_t *message_event)
.....
////////////////////////////////////////////////////////////////////////
        if (!list) {
                //yhy2013-04-04 漏消息记录
                int res;
                if(body && udp_Listen(8387)==0)
                {
                        char tmp[2050];
                        snprintf(tmp,2048, "0\r\n%s\r\n%s\r\n%s",from,to,body);
                        res=udp_SendMsg("127.0.0.1",8388,tmp,strlen(tmp));
                }

                switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR,
"Chat proto [%s]\nfrom [%s]\nto [%s]\n%s\nNobody to send to: Profile %s,udp_send res=%d\n", proto, from, to,
body ? body : "[no body]", prof ? prof : "NULL",res);
```

```
                goto end;
        }
        else
        {
            //yhy2013-04-04 对方收到的所有的消息记录
            int res;
            if(body && udp_Listen(8387)==0)
            {
                char tmp[2050];
                snprintf(tmp,2048, "1\r\n%s\r\n%s\r\n%s",from,to,body);
                res=udp_SendMsg("127.0.0.1",8388,tmp,strlen(tmp));
            }
        }
    ////////////////////////////////////////////////////////////////////////////////
        for (m = list->head; m; m = m->next) {

            if (!(dst = sofia_glue_get_destination(m->val))) {
                switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_CRIT, "Memory Error!\n");
                break;
            }

        if (s_mwi_event) {
                switch_event_fire(&s_mwi_event);
            }
    ……….
```

其次

当用户注册上线的时候发送注册消息：

修改 sofia_reg.c 文件里面的函数：

修改如下函数：

```
uint8_t sofia_reg_handle_register(nua_t *nua, sofia_profile_t *profile, nua_handle_t *nh,
sip_t const *sip,sofia_dispatch_event_t *de, sofia_regtype_t regtype, char *key,
   uint32_t keylen, switch_event_t **v_event, const char *is_nat)
 ……..

        //yhy2013-04-10 add send leak    chat and    call message to user
        if(exptime>0)//>0reg ==0 unreg
        {
            send_leak_chat_message_call=1;
        }

        switch_goto_int(r, 1, end);
```

```
        }


    end:
        switch_safe_free(dup_mwi_account);

        if (auth_params) {
            switch_event_destroy(&auth_params);
        }

        if(send_leak_chat_message_call)
        {
            switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "yhy register ok
exptime=%d,from:%s,to:%s\n", exptime,from_user,to_user);
            if(udp_Listen(8387)==0)
            {
                char tmp[2050];
                snprintf(tmp,2048, "2\r\n%s\r\n%s\r\n%s",from_user,to_user,to_user);
                udp_SendMsg("127.0.0.1",8388,tmp,strlen(tmp));
            }

        }
        return (uint8_t) r;
}
.....................
```

4）接下来是 IM-SERVER 收到 FreeSwitch 发过来的 udp 的离线消息，写入数据库 ：
表结构如下：

```
 //漏话 漏消息表
create table CC_Leak_Task
(
   ID            NUMBER(38) not null,
   Source        VARCHAR2(64) not null,
   Target        VARCHAR2(64) not null,
   Type          NUMBER(2) default 0 not null,--0 call ,1 chat
   State         NUMBER(2) default 0 not null,--0 not send,1 send
   Recvdate      DATE default sysdate not null,
   Data          VARCHAR2(2048) default '' not null,
   Created       DATE default sysdate not null
)

//消息日志表
create table CC_Chat_Log
```

```
(
   ID             NUMBER(38) not null,
   Source         VARCHAR2(64) not null,
   Target         VARCHAR2(64) not null,
   Type           NUMBER(2) default 0 not null,--0 call ,1 chat
   State          NUMBER(2) default 0 not null,--0 not send,1 send
   Recvdate       DATE default sysdate not null,
   Data           VARCHAR2(2048) default '' not null,
   SendDate       DATE default sysdate not null,
   Created        DATE default sysdate not null
);
```

下面是 c 的 code。

```
while(true)
      {
            task=0;
            tmp[0]=0;
            if(udp_WaitEvt(&id, &s_event, 0)==0)      //等待 SOCKET 事件
            {
                  switch(s_event)
                  {
                  case FD_READ://接收到操作，进行处理
                        memset(buffer,0,sizeof(buffer));
                        udp_RecvMsg(ip,buffer,4000);
                        disp_msg("udp_recv:ip=%s,type=%s",ip,buffer);
                        check_udp_message(buffer);
                        break;
                  default:
                        break;
                  }
            }
。。。。。。

int check_udp_message(char*buffer)
{
      char *p,*p2,tmp[4096],from[128],to[128],data[4096];
      int res;

      if(buffer==NULL || buffer[0]==0) return 0;
      p=buffer;
      int i=0;
      tmp[i]=0;
      while(1)
      {
            if(*p==0) break;
```

```
        if(*p=='\r') break;
        if(*p=='\n') break;
        tmp[i++]=*p;
        p++;
    }
    tmp[i]=0;
    int leak_type=atoi(tmp);

///  disp_msg("leak_type=%d",leak_type);

    if(*p==0) return 0;

    p2=p;
    while(1)
    {
        p2++;
        if(*p2==0) break;
        if(*p2=='\r') continue;
        if(*p2=='\n') continue;
        break;
    }
    if(*p2==0) return 0;


    p=p2;
    i=0;
    tmp[i]=0;
    while(1)
    {
        if(*p==0) break;
        if(*p=='\r') break;
        if(*p=='\n') break;
        tmp[i++]=*p;
        p++;
    }
    tmp[i]=0;
    strtok(tmp,"@");

    if(p2=strstr(tmp,"<sip:"))
        strcpy(from,p2+5);
    else
        strcpy(from,tmp);

    //disp_msg("from=%s",from);
```

```
	if(*p==0) return 0;
	p2=p;
	while(1)
	{
		p2++;
		if(*p2==0) break;
		if(*p2=='\r') continue;
		if(*p2=='\n') continue;
		break;
	}
	if(*p2==0) return 0;

	p=p2;
	i=0;
	tmp[i]=0;
	while(1)
	{
		if(*p==0) break;
		if(*p=='\r') break;
		if(*p=='\n') break;
		tmp[i++]=*p;
		p++;
	}
	tmp[i]=0;
	strtok(tmp,"@");
	if(p2=strstr(tmp,"<sip:"))
		strcpy(to,p2+5);
	else
		strcpy(to,tmp);
//	disp_msg("to=%s",to);
	if(*p==0) return 0;
	p2=p;
	while(1)
	{
		p2++;
		if(*p2==0) break;
		if(*p2=='\r') continue;
		if(*p2=='\n') continue;
		break;
	}
	if(*p2==0) return 0;
	p=p2;
	while(1)
```

```
        {
            p2++;
            if(*p2==0) break;
            if(*p2==0x27) {*p2=0x22; continue;}//'==>"
        }
        strcpy(data,p);
        if(leak_type==0)//漏消息提示   写入数据库，同时通知发送者：对方不在线，消息已经存储
        {
            sprintf(tmp,"insert                    into                    CC_Leak_task(id,Source,Target,Type,State,Data)
values(CC_Leak_Task_ID.Nextval,'%s','%s',1,0,'%s')",from,to,data);
        res=ExecSelectSql(tmp,0,0);
        if( strcmp(data,"绯荤粺鎻愮ず:娑堟伅宸茬粡閫佽揪")==0 //系统提示:消息已经送达
||
                strcmp(data,"绯荤粺鎻愮ず:瀵规柟涓嶅湪绾匡紝娑堟伅宸茬粡瀛樺偍")==0||
//系统提示:对方不在线，消息已经存储
                strcmp(data,"绯荤粺鎻愮ず:宸茬粡涓婄嚎")==0)//系统提示:对方上线
                return 0;

            if(send_report&2) send_stateReport_message(to,from,"绯荤粺鎻愮ず:瀵规柟涓嶅湪绾匡紝娑堟伅宸茬粡
瀛樺偍"); //系统提示:对方不在线，消息已经存储

        }
        if(leak_type==1)//送达的记录
        {
            if( strcmp(data,"绯荤粺鎻愮ず:娑堟伅宸茬粡閫佽揪")==0 ||//系统提示:消息已经送达
            strcmp(data,"绯荤粺鎻愮ず:瀵规柟涓嶅湪绾匡紝娑堟伅宸茬粡瀛樺偍")==0||//系统提示:对方不在线，
消息已经存储
                strcmp(data,"绯荤粺鎻愮ず:宸茬粡涓婄嚎")==0) //系统提示:对方上线
                return 0;
            if(strstr(data,"  绯荤粺鎻愮ず:娑堟伅宸茬粡閫佽揪")) return 0;
            if(strstr(data,"  鍛煎彨杩囦綘."))//  呼叫过你.
            {
                if(send_report&2)  send_stateReport_message(to,from,"绯荤粺鎻愮ず:宸茬粡涓婄嚎");//系统提示:
对方上线
                return 0;
            }
        }
        if(leak_type==2)//user register
        {
            check_send_leak_message(from);
        }
        return 0;
}
```

上面的乱码是因为要发送的是 UTF-8 格式的汉字。

5）最后是 IM-SERVER 通过 ESL 接口发送离线消息给刚刚注册上来的用户：

下面是 c 的 code。

```
int send_chat_message(const char*from,const char*to,const char*chatmessage)
{
    int res=0,result=0;
    char tmp[4096];
    ESLevent *e;
    sprintf(tmp,"%d",FreeSwitchport);
    if(con==NULL || con->connected()==false ) con= new ESLconnection(FreeSwitchip, tmp, password);
    e = new ESLevent("custom", "SMS::SEND_MESSAGE");
    sprintf(tmp,"%s\@%s",from,FreeSwitchchatserverip);
    e->addHeader("from", tmp);
    sprintf(tmp,"%s\@%s",to,FreeSwitchchatserverip);
    e->addHeader("to", tmp);
//没有设置这个 text/plain 不行有些软电话会收到消息不提示比如 eyebeam linphone
    e->addHeader("type", "text/plain");
    e->addHeader("sip_profile", "internal");
    e->addHeader("dest_proto", "sip");
    e->addBody(chatmessage);
    con->sendEvent(e);
    disp_msg("from[%s]==>to[%s],data=[%s]",from,to,chatmessage);
    delete e;
    return 0;
}
```

146. 如何利用 FreeSwitch 支持漏话通知？

有了上面的基础，通过修改拨号计划，漏话（呼叫的分机不在线）的时候通过 **ESL** 转到 **IVR** 上，**IVR** 语音通知呼叫者“你呼叫的用户暂时无法接通”，然后把呼叫者和被呼叫者信息写到表里面。将来被呼叫的分机上线了，**IM** 通知之前呼叫者。

归纳如下：当被叫没有注册在线的时候，主叫呼叫被叫，通过 esl 转到
lvr 上，ivr 写入数据库。

拨号计划如下：

```
<extension name="Local_Extension2">
        <condition field="destination_number" expression="^([0-9]\d+)$">
            <action application="export" data="dialed_extension=$1" />
                <action application="set" data="call_timeout=30" />
             <action application="set" data="record_sample_rate=8000" />
                 <action application="export" data="RECORD_STEREO=false" />
                     <action application="set" data="hangup_after_bridge=true" />
                <action application="set" data="continue_on_fail=true" />
<action application="bridge" data="user/${dialed_extension}@${domain_name}" />
```

```
        <action application="socket" data="127.0.0.1:8084 async full"/>
          </condition>
    </extension>
```

C 代码如下：

```
if(strncmp(called,system_called,strlen(system_called))!=0)//被叫号码不是系统号码：漏话
    {
        printf_win32("leak call in,caller=%s,called=%s,sid=%s,ring_connect=%d\n",
        caller,called,sid,ring_connect);
        sprintf(tmp,"insert into CC_Leak_task(id,Source,Target,Type,State,Data)
        values(CC_Leak_Task_ID.Nextval,'%s','%s',0,0,'call you')",caller,called);
        res=ExecSelectSql(tmp,0,0);
        res=CTIPlayDtmf(sid,"hello_leak","",0,0,0,dtmf);
        CTIDropCall(sid);
        return res;
    }
```

将来被叫注册上线的时候，IM 通知被叫：【xxx 在 xxxxxx 时间呼叫过你】
同时 IM 通知主叫：【被叫 XXX 已经上线。】

147. FreeSwitch 离线消息通知和漏话消息通知为什么会有多次？

实际应用中上面的离线消息和漏话消息通知是在用户注册上线的时候系统通知的。
在某些情况下，会给某些软电话发送多次通知，通过抓包之类的分析：
是因为某些软电话在为了支持 NAT 穿透，在启动的时候会注册多次。一次是本地的 ip 上去。
一次是 NAT 之后的 IP 上去，以 pjsip 为例，默认情况下启动会注册两次，如下图：

```
sofia status profile internal reg 1031


Registrations:
=================================================================================================
Call-ID:        84023609ca5d4c9c805a6f7c70c1644f
User:           1031@110.80.10.164
Contact:        "user" <sip:1031@192.168.1.27:8060;ob;fs_nat=yes;fs_path=sip%3A1031%4027.154.234.78%3A8076%3Bob>
Agent:          PJSUA v1.16.0 win32-6.1/i386/msvc-12.0
Status:         Registered(UDP-NAT)(unknown) EXP(2013-06-28 09:11:26) EXPSECS(30)
Host:           freeswitch
IP:             27.154.234.78
Port:           8076
Auth-User:      1031
Auth-Realm:     110.80.10.164
MWI-Account:    1031@110.80.10.164

Call-ID:        84023609ca5d4c9c805a6f7c70c1644f
User:           1031@110.80.10.164
Contact:        "user" <sip:1031@27.154.234.78:8076;transport=UDP;ob>
Agent:          PJSUA v1.16.0 win32-6.1/i386/msvc-12.0
Status:         Registered(UDP)(unknown) EXP(2013-06-28 09:11:28) EXPSECS(32)
Host:           freeswitch
IP:             27.154.234.78
Port:           8076
Auth-User:      1031
Auth-Realm:     110.80.10.164
MWI-Account:    1031@110.80.10.164

Total items returned: 2
=================================================================================================
```

其中第一次的注册实际上是虚的，后续的注册

假如 pjsip 的 expire time 设置 30 秒过了 60 秒再看，就变成了一个注册在线了：

```
sofia status profile internal reg 1031


Registrations:
=================================================================================================
Call-ID:        84023609ca5d4c9c805a6f7c70c1644f
User:           1031@110.80.10.164
Contact:        "user" <sip:1031@27.154.234.78:8076;transport=UDP;ob>
Agent:          PJSUA v1.16.0 win32-6.1/i386/msvc-12.0
Status:         Registered(UDP)(unknown) EXP(2013-06-28 09:11:53) EXPSECS(12)
Host:           freeswitch
IP:             27.154.234.78
Port:           8076
Auth-User:      1031
Auth-Realm:     110.80.10.164
MWI-Account:    1031@110.80.10.164

Total items returned: 1
=================================================================================================
```

要解决这个有两个办法一个是修改 FreeSwitch：

<param name="multiple-registrations" value="false"/>

改为不允许多个软电话使用一个帐号注册，以实现共振，但是有些场景又要允许，这样只能修改软电话上的了。

另外一个是修改软电话上的参数，比如 pjsip，修改参数：

--auto-update-nat=N Where N is 0 or 1 to enable/disable SIP traversal behind
symmetric NAT (default 1)

启动的时候 带上 --auto-update-nat=0 参数就可以。Pjsip 启动之后参看，就只有一个注册了：

```
sofia status profile internal reg 1031


Registrations:
=================================================================================================
Call-ID:        3c1cc879d6754161881b5a43d52822e9
User:           1031@110.80.10.164
Contact:        "user" <sip:1031@192.168.1.27:8060;ob;fs_nat=yes;fs_path=sip%3A1031%4027.154.234.78%3A8076%3Bob>
Agent:          PJSUA v1.16.0 win32-6.1/i386/msvc-12.0
Status:         Registered(UDP-NAT)(unknown) EXP(2013-06-28 09:13:57) EXPSECS(29)
Host:           freeswitch
IP:             27.154.234.78
Port:           8076
Auth-User:      1031
Auth-Realm:     110.80.10.164
MWI-Account:    1031@110.80.10.164

Total items returned: 1
=================================================================================================
```

148. FreeSwitch 如何支持 IM 聊天中对方状态功能？

本人测试过 linphone 的 windows 版本和 FreeSwitch 配合，支持对方状态显示功能。

实现的理论基础如下: http://www.ietf.org/rfc/rfc3265.txt

Session Initiation Protocol (SIP)-Specific Event Notification

消息的订阅和通知事件。

订阅和通知的基本理念是：在网络中的实体可以订阅在网络中各种不同资源和呼叫的资源和呼叫状态，并且那些实体当状态发生变化时会发送通知消息。

一个典型的呼叫流程如下图所示：

订阅者　　通知者
|-----SUBSCRIBE---->| 请求状态订阅
|<---------200----------| 订阅请求响应
|<------NOTIFY---------| 返回当前状态通知
|--------200---------->|
|<------NOTIFY---------| 返回当前状态通知
|--------200---------->|
订阅消息是会过期的，必须由后续的 SUBSCRIBE 来刷新。在 linphone 和 FreeSwitch 具体写作过程描述如下：
1）.分机 1526 的 linphone 启动的时候，先注册到 FreeSwitch 上，注册通过之后，发 SUBSCRIBE 到 FreeSwitch，订阅某个分机的状态，如下图：订阅 1527 的分机的状态，SUBSCRIBE 的超时时间是 600 秒，订阅成功之后 FreeSwitch 马上就会发送 1527 的当前状态给 1526 的 linphone

9	0.244991	192.168.1.101	110.80.10.164	SIP	Request: REGISTER sip:110.80.10.164
10	0.336857	110.80.10.164	192.168.1.101	SIP	Status: 200 OK (1 bindings)
11	3.082394	192.168.1.101	110.80.10.164	SIP	Request: SUBSCRIBE sip:1527@110.80.10.164
12	3.151066	110.80.10.164	192.168.1.101	SIP	Status: 202 Accepted
13	3.155752	110.80.10.164	192.168.1.101	SIP	Request: NOTIFY sip:1526@121.11.252.68
14	3.156808	192.168.1.101	110.80.10.164	SIP	Status: 200 OK
15	18.213000	192.168.1.101	110.80.10.164	UDP	Source port: 5060 Destination port: 5060

```
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol
  ⊞ Request-Line: SUBSCRIBE sip:1527@110.80.10.164 SIP/2.0
  ⊟ Message Header
      Via: SIP/2.0/UDP 192.168.1.101:5060;rport;branch=z9hG4bK18089
    ⊞ From: <sip:1526@110.80.10.164>;tag=575
    ⊞ To: "test" <sip:1527@110.80.10.164>
      Call-ID: 32281
      CSeq: 20 SUBSCRIBE
    ⊞ Contact: <sip:1526@121.11.252.68>
      Max-Forwards: 70
      User-Agent: Linphone/3.5.2 (eXosip2/3.6.0)
      Expires: 600
      Event: presence
      Content-Length: 0
```

2）将来当分机 1527 状态改变的时候，比如 1527 分机下线的时候，FreeSwitch 会通过 NOTIFY 方法通知给分机 1526linphone，告知 1527 分机的状态变化，抓包如下图；

```
    1 0.000000    110.80.10.164     192.168.1.101     SIP    Request: NOTIFY sip:1526@121.11.252.68:6176
    2 0.001023    192.168.1.101     110.80.10.164     SIP    Status: 200 OK
```

```
⊞ Frame 1 (1310 bytes on wire, 1310 bytes captured)
⊞ Ethernet II, Src: 14:d6:4d:e1:7f:58 (14:d6:4d:e1:7f:58), Dst: 84:a6:c8:7b:b2:40 (84:a6:c8:7b:b2:40)
⊞ Internet Protocol, Src: 110.80.10.164 (110.80.10.164), Dst: 192.168.1.101 (192.168.1.101)
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol
  ⊞ Request-Line: NOTIFY sip:1526@121.11.252.68:6176 SIP/2.0
  ⊞ Message Header
  ⊟ Message body
      <?xml version="1.0" encoding="ISO-8859-1"?> \n
      <presence xmlns='urn:ietf:params:xml:ns:pidf' \n
      xmlns:dm='urn:ietf:params:xml:ns:pidf:data-model' \n
      xmlns:rpid='urn:ietf:params:xml:ns:pidf:rpid' \n
      xmlns:c='urn:ietf:params:xml:ns:pidf:cipid' entity='sip:1527@110.80.10.164'>\n
       <tuple id='t6a5ed77e'>\n
        <status>\r\n
         <basic>closed</basic>\n
        </status>\n
       </tuple>\n
       <dm:person id='p06360c4a'>\n
        <dm:note>Unregistered</dm:note>\n
       </dm:person>\n
      </presence>
```

149. FreeSwitch 的 IM 聊天中对方状态存在啥问题？

a) 状态刷新不及时的问题:
linphone 的 windows 版本和 FreeSwitch 的配合状态显示功能，存在在用户异常断开的时候状态刷新不及时的问题。
究其原因：
1 是由于 linphone 的参数的限制导致的。
2 是由于 FreeSwitch 代码设计的考虑(具体原因偶尚未搞清楚)。

b) linphone 在屏幕没有亮的情况下 register time 会变成 120 秒左右，
即使超时时间 expire 参数设置为 30 秒也没有用

c) linphone 退出对方看到状态及时变成了离线状态，但是 linphone 重新启动之后，
对方看到的状态依然是离线状态.
要等多久才会变呢？根据 linphone 参数配置，间隔时间是 600 秒。
另外，根据测试是有这个分机呼叫或者被呼叫，状态才会通知过来。
这个可以修改 linphone 或者修改 FreeSwitch。
办法是：
修改 Linphone：默认情况下发送订阅的间隔时间是 600 秒，可以修改为每次注册
都重新发订阅消息。
修改 FreeSwitch：在用户注册上来之后，通知相应的订阅者。

d) linphone windows 版本上加了联系人以后呼叫，安卓版本上看到的呼叫历史是分机号码，没有 ip 地址(这个不是问题，看起来不一致)

150. FreeSwitch 的 IM 聊天中对方状态刷新延时如何优化？

上面说在用户异常断开的时候，存在状态刷新延时的问题。

有两个地方可以优化：

1） FreeSwitch 上的优化：这个又得修改源码！

FreeSwitch 会把软电话注册上来的 expire 注册超时时间加上 60 秒，然后写到数据库里面，这样就导致用户异常断开的时候，至少多等 60 秒才会通知订阅的分机。

具体修改的地方如下：

修改**sofia_reg.c** 里面的

uint8_t sofia_reg_handle_register(nua_t *nua, sofia_profile_t *profile, nua_handle_t *nh, sip_t const *sip,sofia_dispatch_event_t *de, sofia_regtype_t regtype, char *key, uint32_t keylen, switch_event_t **v_event, const char *is_nat)

函数里面的

```
switch_find_local_ip(guess_ip4, sizeof(guess_ip4), NULL, AF_INET);

        contact = sofia_glue_get_url_from_contact(contact_str, 1);
        url = switch_mprintf("sofia/%q/sip:%q", profile->name, sofia_glue_strip_proto(contact));

        switch_core_add_registration(to_user, reg_host, call_id, url, (long) reg_time + (long) exptime + YHY_EXP_TIMER,
                                                network_ip, network_port_c, is_tls ? "tls" : is_tcp ? "tcp" : "udp",
reg_meta);

        switch_safe_free(url);
        switch_safe_free(contact);

        if (!update_registration) {
            sql = switch_mprintf("insert into sip_registrations "
                        "(call_id,sip_user,sip_host,presence_hosts,contact,status,rpid,expires,"

 "user_agent,server_user,server_host,profile_name,hostname,network_ip,network_port,sip_username,sip_realm,"
                        "mwi_user,mwi_host, orig_server_host, orig_hostname, sub_host) "
                        "values ('%q','%q', '%q','%q','%q','%q', '%q', %ld, '%q', '%q', '%q', '%q', '%q', '%q',
'%q','%q','%q','%q','%q','%q','%q','%q')",
                        call_id, to_user, reg_host, profile->presence_hosts ? profile->presence_hosts : "",
                        contact_str, reg_desc, rpid, (long) reg_time + (long) exptime + YHY_EXP_TIMER,
                        agent, from_user, guess_ip4, profile->name, mod_sofia_globals.hostname, network_ip,
network_port_c, username, realm,
                                                mwi_user, mwi_host, guess_ip4, mod_sofia_globals.hostname, sub_host);
        } else {
            sql = switch_mprintf("update sip_registrations set call_id='%q',"
                                                "sub_host='%q', network_ip='%q',network_port='%q',"
                                                "presence_hosts='%q', server_host='%q', orig_server_host='%q',"
```

```
                                        "hostname='%q', orig_hostname='%q',"
                    "expires = %ld where sip_user='%q' and sip_username='%q' and sip_host='%q' and contact='%q'",
                                        call_id, sub_host, network_ip, network_port_c,
                                    profile->presence_hosts ? profile->presence_hosts : "", guess_ip4, guess_ip4,
                                        mod_sofia_globals.hostname, mod_sofia_globals.hostname,
                                    (long) reg_time + (long) exptime + YHY_EXP_TIMER,
                                    to_user, username, reg_host, contact_str);
        }

        if (sql) {
            sofia_glue_execute_sql_now(profile, &sql, SWITCH_TRUE);
        }
```

.....

… 其中 YHY_EXP_TIMER原来是60，现在改为一个宏定义，设置为1.

2） linphone 上的优化修改：

linphone 的注册的 expire 参数最小不能小于 30 秒，即使设置为 2 秒，linphone 发出去的还是 30 秒，修改源码，改为最小不能小于 2 秒。

这样本分机用户异常断开的时候，其它分机用户 3 秒（这个地方的 2 秒加上上面 FreeSwitch 的 1 秒）就会收到用户下线消息，否则就要等 30+60=90 秒。

3） 修改 FreeSwitch：在用户注册上来之后，通知相应的订阅者。

修改

```
uint8_t sofia_reg_handle_register(nua_t *nua, sofia_profile_t *profile, nua_handle_t *nh, sip_t const *sip,
                                  sofia_dispatch_event_t *de, sofia_regtype_t regtype, char *key,
                                  uint32_t keylen, switch_event_t **v_event, const char *is_nat)
{
.....
end:
 switch_safe_free(dup_mwi_account);
 if (auth_params) {
        switch_event_destroy(&auth_params);
 }
 if(exptime>0)//>0reg ==0 unreg
{
  //yhy2013-04-23 用户注册上来通知其它订阅的用户，用户状态由离线改为在线。
        if (switch_event_create(&sevent, SWITCH_EVENT_PRESENCE_IN) == SWITCH_STATUS_SUCCESS) {
            switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "proto", SOFIA_CHAT_PROTO);
            switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "rpid", rpid);
            switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "login", profile->url);
            switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "user-agent",
                                           (sip && sip->sip_user_agent) ? sip->sip_user_agent->g_string :
```

```
"unknown");
                snprintf(tmp,2048, "%s@%s",to_user,sub_host);
                switch_event_add_header(sevent, SWITCH_STACK_BOTTOM, "from", tmp);//"%s@%s", to_user, sub_host);
                switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "status", "Registered");
                switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "presence-source", "register");
                switch_event_add_header_string(sevent, SWITCH_STACK_BOTTOM, "event_type", "presence");
                switch_event_fire(&sevent);
        }
 }
 return (uint8_t) r;
}
....
```

其它说明：这个通过注册时间的办法实现代价比较大，在在线用户量到的时候可能会有瓶颈，要考虑进行优化或者使用其它替代方案。

第九章 FreeSwitch 媒体部分

FreeSwitch 的媒体操作可以通过 ESL 来实现，也可以简单通过拨号计划实现。
这个部分介绍的是通过拨号计划实现的简单控制。

151. FreeSwitch 默认播音和录音目录在哪里？

播放文件的默认目录 ：
\sounds\en\us\callie
录音文件的默认目录:
\sounds\en\us\callie

152. FreeSwitch 如何指定录放音文件和目录？

可以在 play 或者 record 的时候写上绝对路径
比如 d:/record/1.wav

153. FreeSwitch 如何指定 alaw 播音文件的格式？

FreeSwitch 的播音除 wav 文件之外，其它是根据文件扩展名来区分格式的。
国内通常的 ivr 系统都是使用单声道的 8000Hz alaw 编码的语音格式
要做到然 FreeSwitch 能支持这种格式的语音有两种办法：
A. 文件名称的扩展名叫做 .alaw
B. 使用工具给这些文件加上 wav 头，然后 扩展名叫做 .wav
假如 是其它格式的语音类似参考修改。
具体可以在 FS_Cli 下使用 show files 命令查看。然后使用合适的语音文件扩展名

154. FreeSwitch 如何指定 alaw 录音文件的格式？

国内通常的 ivr 系统都是使用单声道的 8000Hz alaw 编码的语音格式
要做到能录制到这种格式的语音：
需要以下几个步骤：

先在拨号计划 conf\dialpla\default.xml 里面加上：

```
<action application="set" data="record_sample_rate=8000"/>
<action application="export" data="RECORD_STEREO=false"/>
```

来指定是 8000HZ 和单声道。默认是立体声的。

比如：

```
<extension name="socket">
        <condition field="destination_number" expression="^12396$">
        <action application="set" data="record_sample_rate=8000"/>
        <action application="export" data="RECORD_STEREO=false"/>
        <action application="socket" data="127.0.0.1:8084 async full"/>
        </condition>
        </extension>
```

然后：
在 uuid_record 或者 record 的时候
录音的时候指定是 .alaw 的扩展名，这样就指定是 alaw 编码的语音。
假如是.wav 扩展名。则是线性 16bit 没有编码的语音。

155. FreeSwitch 如何实现给拨入的电话播音？

以国内通常使用的单声道的 8000Hz pcm alaw 格式 语音为例：
拨号计划 conf\dialpla\default.xml 里面 在 <context name="default"> 后面加上：

```
<extension name="Local_Extension2">
 <condition field="destination_number" expression="^([0-9]\d+)$">
      <action application="export" data="dialed_extension=$1"/>
      <action application="set" data="call_timeout=30"/>
       <action application="set" data="record_sample_rate=8000"/>
       <action application="export" data="RECORD_STEREO=false"/>
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=true"/>

      <action application="answer"/>
      <action application="sleep" data="1000"/>
      <action application="playback" data="d:/data/system/zzy_hello.alaw"/>
      <action application="hangup"/>
 </condition>
 </extension>
```

zzy_hello.alaw 也可以是带 wav 头的语音。

156. FreeSwitch 如何实现对拨入的电话录音？

以单声道的 8000Hz pcm alaw 格式 语音为例：

拨号计划 conf\dialpla\default.xml 里面 在 <context name="default"> 后面加上：

```
<extension name="Local_Extension2">
 <condition field="destination_number" expression="^([0-9]\d+)$">
      <action application="export" data="dialed_extension=$1"/>
      <action application="set" data="call_timeout=30"/>
       <action application="set" data="record_sample_rate=8000"/>
       <action application="export" data="RECORD_STEREO=false"/>
      <action application="set" data="hangup_after_bridge=true"/>
           <action application="set" data="continue_on_fail=true"/>

      <action application="answer"/>
      <action application="sleep" data="1000"/>
      <action application="playback" data="d:/data/system/zzy_hello.alaw"/>
      <action application="set" data="playback_terminators=#"/>
      <action application="record"
    data="z:/record/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_${destination_number}_${caller_id_nu
  mber_${uuid}}.alaw  60 "/>
      <action application="hangup"/>
  </condition>
    </extension>
```

说明：

.alaw 表示 pcm alaw 格式语音，

假如要录制 MP3 格式的语音， 直接使用 .mp3 作为后缀就可以，FreeSwitch 使用文件的扩展名进行格式的识别。

60 表示录制 60 秒。

${strftime(%m%d)} 表示 月日 表示每一天一个目录。

${strftime(%Y-%m-%d-%H-%M-%S)} 表示年月日 时分秒

${destination_number} 是被叫号码

${caller_id_number} 是主叫号码

${uuid} 是每个通道的唯一变量

使用这些作为文件名 是为了保证 录音的文件不重复

说明：

```
<action application="record"
data="z:/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_${destination_number}_
${caller_id_number}.alaw 20 200"/>
```

record 命令假如文件目录不存在，FreeSwitch 不会自动创建文件目录。

157. FreeSwitch 如何实现对通话的双方全程录音？

以单声道的 8000Hz pcm alaw 格式 语音为例：

拨号计划 conf\dialpla\default.xml 里面 在 <context name="default"> 后面加上：

```
<extension name="Local_Extension2">
 <condition field="destination_number" expression="^([0-9]\d+)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="set" data="call_timeout=30"/>
         <action application="set" data="record_sample_rate=8000"/>
         <action application="export" data="RECORD_STEREO=false"/>
        <action application="set" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=true"/>
        <action application="export" data="execute_on_answer=record_session
z:/record/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_${caller_id_number}_${destination_number}.alaw"/
>
 <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
 </condition>
</extension>
```

说明：

<action application="export" data="execute_on_answer=record_session z:/record/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_${caller_id_number}_${destination_number}.alaw"/>

record_session 命令假如文件目录不存在，FreeSwitch 会自动创建文件目录。

158. FreeSwitch 如何实现对拨入参加会议？

假设 拨入 8XXX 参加 会议。拨号计划里面可以这样写:

拨号计划 conf\dialpla\default.xml 里面 在 <context name="default"> 后面加上：

```
<extension name="Local_Conf8">
 <condition field="destination_number" expression="^(8\d+)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="set" data="call_timeout=30"/>
         <action application="set" data="record_sample_rate=8000"/>
         <action application="export" data="RECORD_STEREO=false"/>
        <action application="set" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=true"/>
        <action application="answer"/>
        <action application="sleep" data="1000"/>
        <action application="playback" data="d:/data/system/zzy_conf.alaw"/>
        <action application="conference" data="$1@default"/>
        <action application="hangup"/>
 </condition>
   </extension>
```

159. FreeSwitch 如何实现对会议参数控制？

会议的详细控制有两种，一种是在会议的配置文件里控制，一种是通过 esl 进行控制。下面以会议的配置文件里控制为例：

conf\autoload_configs\ conference.conf.xml

```
<caller-controls>
    <group name="default">
      <control action="mute" digits="0"/>
      <control action="deaf mute" digits="*"/>
      <control action="energy up" digits="9"/>
      <control action="energy equ" digits="8"/>
      <control action="energy dn" digits="7"/>
      <control action="vol talk up" digits="3"/>
      <control action="vol talk zero" digits="2"/>
      <control action="vol talk dn" digits="1"/>
      <control action="vol listen up" digits="6"/>
      <control action="vol listen zero" digits="5"/>
      <control action="vol listen dn" digits="4"/>
      <control action="hangup" digits="#"/>
    </group>
</caller-controls>
```

上面这个参数是参加会议成员在会场可以按键控制，比如 按 0 静音
按 6 键 加大听的音量，按 4 键减小听音量之类的。
实际应用中假如不需要用户按键控制，就都注释掉。

下面是一个在用的会议的控制参数：

```
<!--If no profile is specified it will default to "default"-->
<profile name="default">
  <param name="domain" value="$${domain}"/>
  <!--   会议的采样频率，电话信道都是 8000 -->
  <param name="rate" value="8000"/>
  <!--   每帧多少毫秒 20  表示 20 毫秒  -->
  <param name="interval" value="20"/>
  <!--给它人听的时候的最小能量，这个大了会导致说话小声了，别人就听不见-->
  <param name="energy-level" value="0"/>
  <!--会场最大人数-->
  <param name="max-members" value="120"/>
  <!--  被静音了之后的提示音  -->
  <param name="muted-sound" value="conference/conf-muted.wav"/>
  <!--  被解除静音了之后的提示音  -->
  <param name="unmuted-sound" value="conference/conf-unmuted.wav"/>
```

```
        <!-- 会场变成只有一个人的时候的提示音，这个是英文语音需要自己重新录
          制成中文 -->
        <param name="alone-sound" value="conference/conf-alone.wav"/>

        <!-- 会场只有一个人的时候的音乐-->
        <param name="moh-sound" value="$${hold_music}"/>
        <!-- 进入会场的提示音 -->
        <param name="enter-sound" value="tone_stream://%(200,0,500,600,700)"/>
        <!-- 退出会场的提示语音 -->
        <param name="exit-sound" value="tone_stream://%(500,0,300,200,100,50,25)"/>

        <!-- 提示输入密码的提示音 -->
        <param name="pin-sound" value="conference/conf-pin.wav"/>
        <!-- 密码错误提示 -->
        <param name="bad-pin-sound" value="conference/conf-bad-pin.wav"/>
        <!-- 会议密码 -->
        <param name="pin" value="1234"/>

        <!-- 密码最大允许错误次数 -->
        <param name="pin-retries" value="3"/>

      <!-- 当会场没有声音的时候=0 输出完全的静音 -->
        <param name="comfort-noise" value="0"/>

      <!—默认是只有说话的人才被混音，改为所有参与人都混音-->
       <param name="conference-flags" value="audio-always"/>

      <!— 自动增益控制的参数，越大会议声音就越大 =true 使用默认增益 -->
       <param name=" auto-gain-level "    value="1100"/>

        <!-- Uncomment 自动会议录音 -->
        <param name="auto-record"
value="$${recordings_dir}/${conference_name}_${strftime(%Y-%m-%d-%H-%M-%S)}.wav"/>
</profile>
```

说明：会议相关的语音文件在 sounds\en\us\callie\conference\8000 目录下
里面的语音 比如 会场只有一个人的时候的提示语音：conf-alone.wav 都是英文提示，需要根据实际需要进行录制修改替换。

160. FreeSwitch 录音文件回放时发现声音很小如何解决？

录音生成的 WAV 文件，和实际录音对比声音只是小一点点。
使用 cooledit 播放声音是几乎正常的。
回放录音文件时发现声音很小，可以通过设置增益来实现:

```
uuid_audio <uuid> [start [read|write] [mute|level <level>]|stop]
```

level 参数从 -4 to 4, 0 是默认声音大小. 4 是最大, write 表示是用户（分机）听见的，比如，A 和 B 通话，要放大 A 听 B 的声音太小，就在 A 的 uuid 上设置 write，level 设置 4。
设置 play 的时候使用 write，比如播音声音放大到最大：uuid_audio 0019a4f3-7ea1-4b7a-89c2-c2c8205e87e5 start write level 4 实际测试在通话过程中可以设置这个参数来调整软电话的声音音量大小。

161. FreeSwitch 如何实现支持视频的录制和播放？

FreeSwtich 通过 模块 fsv 支持视频的录制和播放，此模块提供两个 app,record_fsv 或者 record 和 play_fsv, 一个录像，一个播放，其实现在目录 src\mod\applications\mod_fsv/mod_fsv.c
使用下面方法录像：
dialplan 中 调用 app record_fsv 或者 record
<action application="record_fsv" data="file.fsv"/>
参数为录音文件名。播放视频：
dialplan 中调用 app play_fsv
<action application="play_ fsv" data="file.fsv"/>
参数同样为 文件名
在 api 里面 使用： uuid_record <uuid> start file.fsv
uuid_record 只能录制一个 leg 的视频。
会场里面的录制： conference 3000 record file.fsv
说明： FreeSwitchv 表示是"FS Video"的视频，里面的格式是 线性 pcm 的语音和视频编码的原始 rtp 包的组合，文件头格式如下：

```
struct file_header {
    int32_t version;
    char video_codec_name[32];
    char video_fmtp[128];
    uint32_t audio_rate;
    uint32_t audio_ptime;
    switch_time_t created;
};
```

帧按顺序保存，开始 4byte 是帧大小。假如第一个 bit 是 0 表示此帧是语音，假如是 1 表示此帧是视频。其它 31bit 才是真正的帧长度,默认配置下，加上视频编码器，使用支持视频的软电话拨打 9193 和 9194 就是录制和播放视频。Fsv 格式的视频只能使用 freeswitch 才能播放。
本人有工具可以实现 fsv 格式转换为 MP4 的文件可以使用 mediaPlay 进行播放，有需要可以联系本人。

162. FreeSwitch 如何实现收软电话发过来的传真？

FreeSwitch 传真的介绍参见：
http://wiki.FreeSwitch.org/wiki/Mod_spandsp#Invoking_the_app_from_the_CLI

要做传真之前需要先了解一下 传真协议：
T.30 和 T.38 是目前常用的传真协议。
T.30 是之前的公用电话交换网 PSTN 上的文件传真协议。ITU 的 T.30 标准文档 T.30 标准定义了三

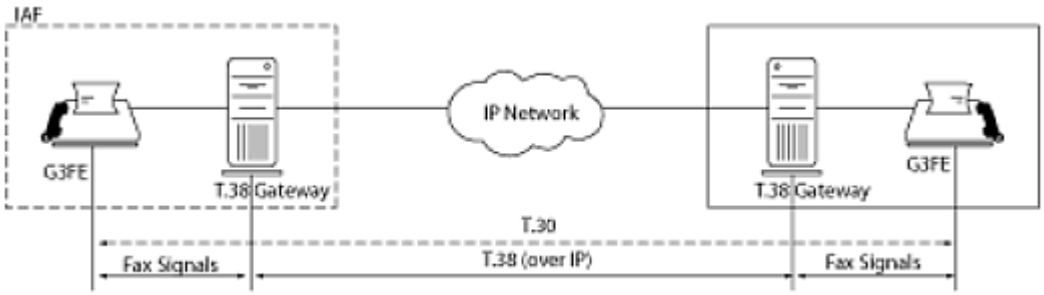
类传真机的传输标准。

T.38 是 IP 传真的标准，由 ITU 第 8 研究组于 1998 年 6 月 18 号批准认可。并定名为"T.38: Procedures for Real-Time Group 3 Facsimile Communication Over IP Networks,"该标准定义了终端之间发送组 3 电传的过程，定义了终端之间传输路径部分，包括 PSTN 、ISDN-一种类似 Internet 的 IP 网络。

T.38 定义了在 Internet 上适用的 T.38 传真设备和 T.38IP 传真网关的 IP 网络协议。在 Internet 上适用的 T.38 传真设备可以与其它的类似设备或 T.38IP 传真网关直接通讯。T.38 传真网关能提供以下功能：

●发送方网关解调输入的 T.30 传真信号
●将 T.30 传真信号转换为 T.38 因特网传真协议（IFP）包
●发送方和接收方的 T.38 网关之间交换 IFP 包
●接收方将 T.38 IFP 包还原成 T.30 信号
●调制 T.30 信号并传送至目标 传真机

T.30 和 T.38
它们的区别可以使用下图来说明：



简单一句话归纳就是 T.30 是之前电话网络 E1 或者模拟线路上的传真协议
T.38 是 IP 网络上的传真协议。
显然 FreeSwitch 的传真是 T.38 协议，但是它也支持通过 voip 网关 或者 T.38 的 gateway 跟传统的支持 T.30 协议的传真机进行收发。

FreeSwitch 的传真 7788 的几个模块组合成 mod_spandsp 模块 ：
mod_fax, mod_t38gateway, and the mod_voipcodecs 等模块
有一个注意的是 T.30 传真需要使用 711 的编码，否则不工作。

先准备测试能收发传真的软电话客户端工具，
本人测试支持 T.38 的软电话 有 kapanga，
下载安装之后试用的授权 30 天之内可以使用传真功能，时间过了就不行了，除非使用非常的手段比如修改软电话机器的时间才行。 :-）
kapanga 下载地址：
http://www.kapanga.net/IP/home.cfm

收传真通常定义为系统收传真。一般情况下是 FreeSwitch 作为被叫收传真。
也就是说通常收传真的定义是用户拨打到 FreeSwitch 上，用户发传真，FreeSwitch 收传真。
（还有一种比较意外比较变态的应用是 FreeSwitch 拨打到用户上，然后用户发传真，FreeSwitch

收传真。我们这里不讨论这个应用。）

然后在拨号计划 conf\dialpla\default.xml 里面 在 <context name="default"> 后面加上：

接收传真拨号计划如下：

```
<extension name="myfax_receive">
 <condition field="destination_number" expression="^(91\d+)$">
  <action application="answer" />
 <action application="set" data="fax_enable_t38=true"/>
 <action application="set" data="fax_enable_t38_request=true"/>
  <action application="playback" data="silence_stream://2000"/>
  <action application="rxfax"
data="z:/fax/${strftime(%m%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_
${destination_number}_${caller_id_number}_${uuid}/rxfax.tif"/>
  <action application="hangup"/>
        </condition>
     </extension>
```

关键是 <action application="set" data="fax_enable_t38=true"/>

<action application="set" data="fax_enable_t38_request=true"/>

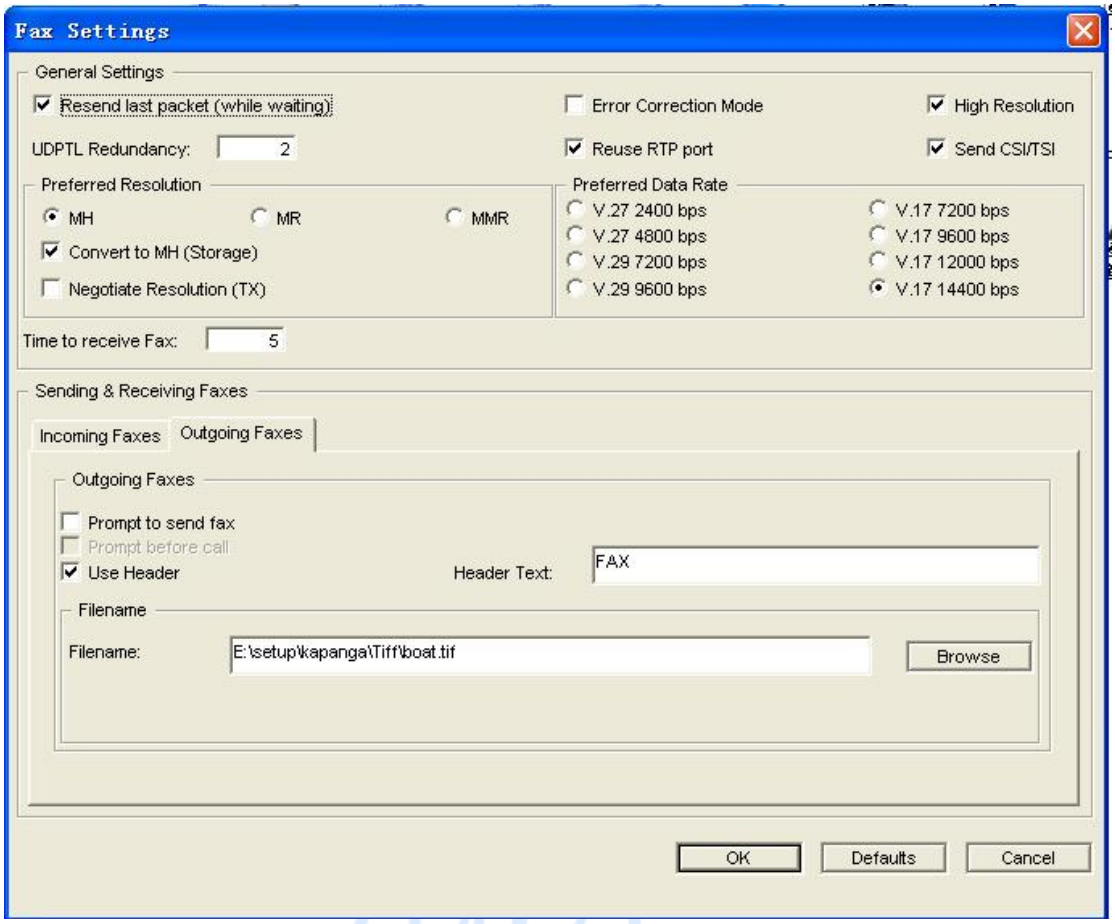
两句，表示 send re-INVITE for T.38 这个是标准要求的。没有这个及时有传真音也握手不上。

收下来是 tif 格式的图片文件。

然后配置 kapanga 使用，

配置使用 ulaw 或者 alaw 语音编码器，去掉视频编码器，去掉摄像头

特别是发传真的配置，默认是发 pdf 文件，本人测试没有安装 pdf 阅读器是不能发送的，需要改为 tif 文件:



然后开始呼叫，呼叫通了，FreeSwitch 货播放 唧唧。。。唧唧。。。的传真握手声音，几秒之后握手成功，kapanga 会自动开始发送传真。kapanga 界面会显示握手信息啥的，握手上去了就开始发送。

界面如下：

FreeSwitch 这边，握手成功，开始接收之后，FreeSwitch 上显示如下信息：

```
。。。
sofia/internal/Username@192.168.1.168:5060 image media sdp:
v=0
o=FreeSwitch 1373951373 1373951377 IN IP4 192.168.1.151
s=FreeSwitch
c=IN IP4 192.168.1.151
t=0 0
m=image 15756 udptl t38
a=T38FaxVersion:0
a=T38MaxBitRate:14400
a=T38FaxFillBitRemoval
a=T38FaxRateManagement:transferredTCF
a=T38FaxMaxBuffer:2000
a=T38FaxMaxDatagram:400
a=T38FaxUdpEC:t38UDPRedundancy

…
2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:332 === Negotiation Result==
2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:333 Remote station id: b?  ?
```

.f 鏊丨嘎 v?喊

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:334 Local station id: SpanDSP Fax Ident

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:335 Transfer Rate: 14400

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:337 ECM status off

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:338 remote country:

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:339 remote vendor:

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:340 remote model:

2013-07-16 17:32:25.765850 [DEBUG] mod_spandsp_fax.c:342 ==========

接收成功之后

FreeSwitch 上显示：

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:443 ==== Page Received ==========

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:444 Page no = 1

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (bi-level in the file)

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 1623 pixels (1728 x 1623 pixels in the file)

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031/m x 7700/m (8031/m x 7700/m in the file)

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 1-D (1)

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 119673 bytes

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 0

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 0

2013-07-16 17:33:39.486067 [DEBUG] mod_spandsp_fax.c:452 ==========

...

2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:511 ==========

2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:517 Fax successfully received.

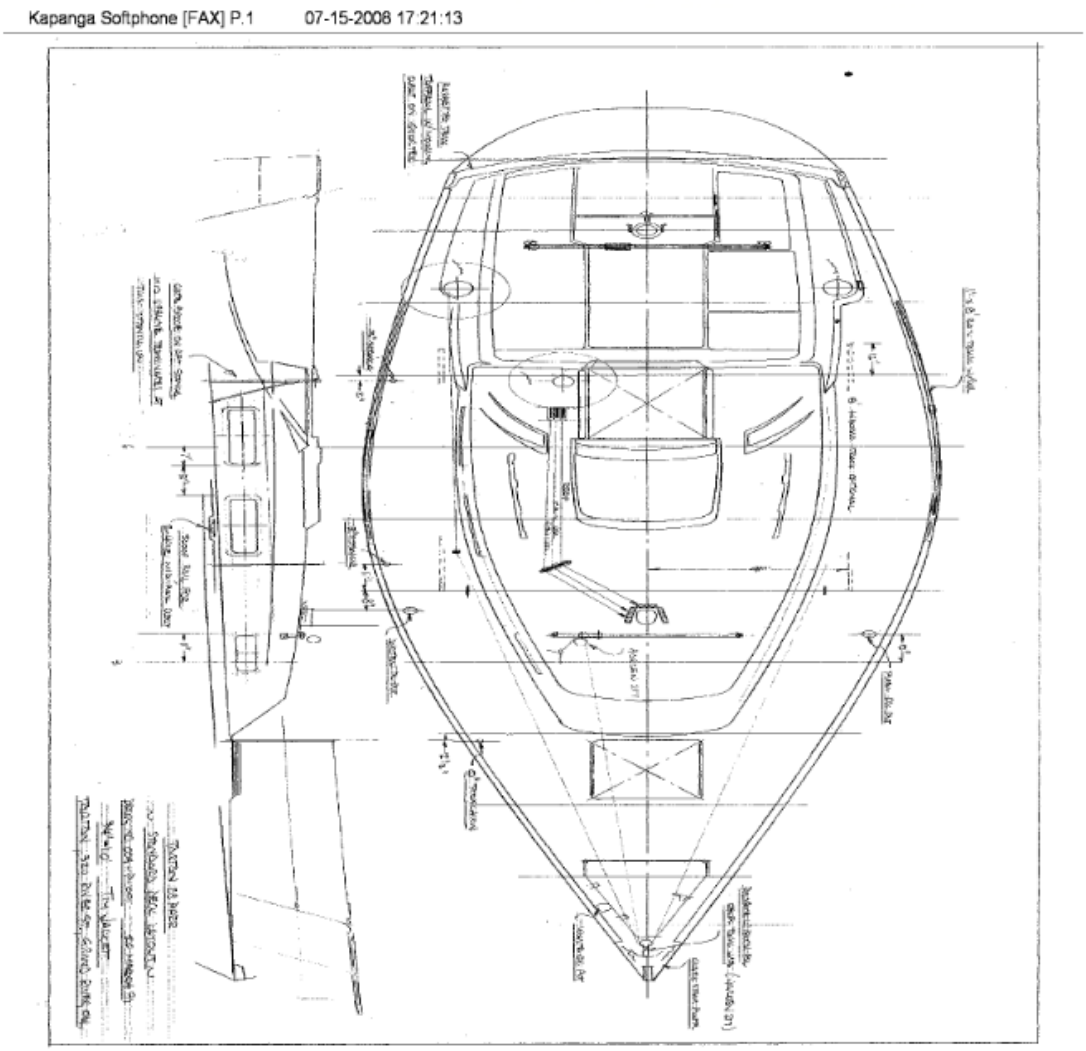
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:529 Remote station id: b? 瑞 ?

.f 鏊」嘆 v?喊

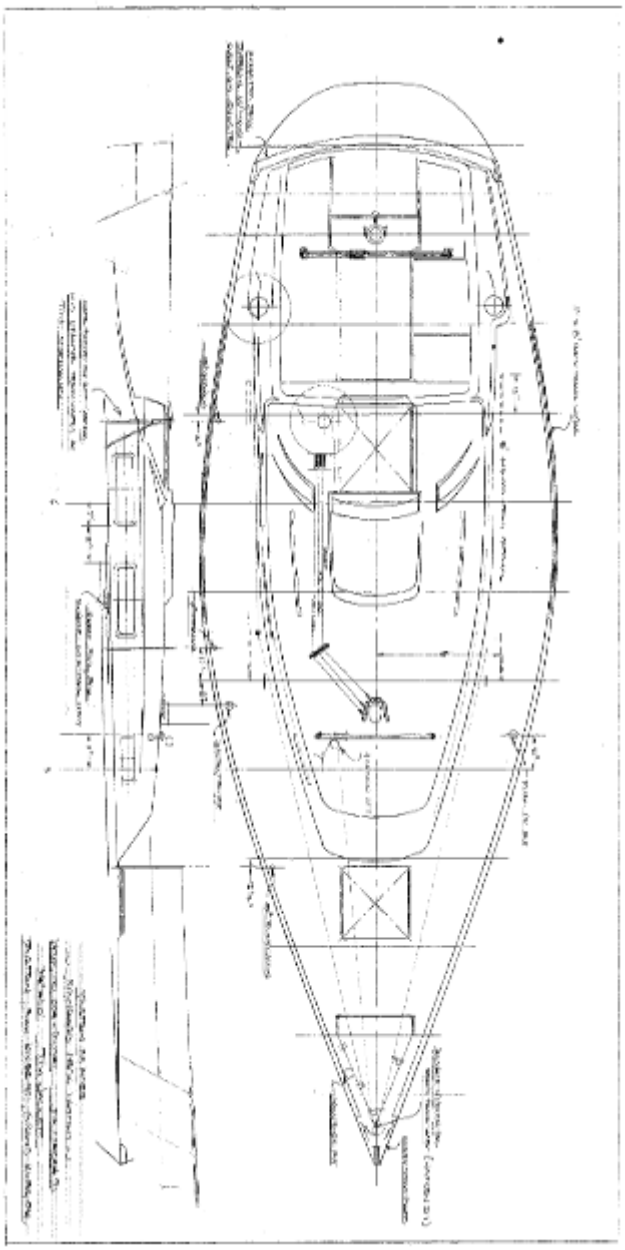
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:530 Local station id: SpanDSP Fax Ident
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:531 Pages transferred: 1
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:533 Total fax pages: 1
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:534 Image resolution: 8031x7700
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:535 Transfer Rate: 14400
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:537 ECM status off
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:538 remote country:
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:539 remote vendor:
2013-07-16 17:33:40.786141 [DEBUG] mod_spandsp_fax.c:540 remote model:

。。。。

收下来的结果如下：



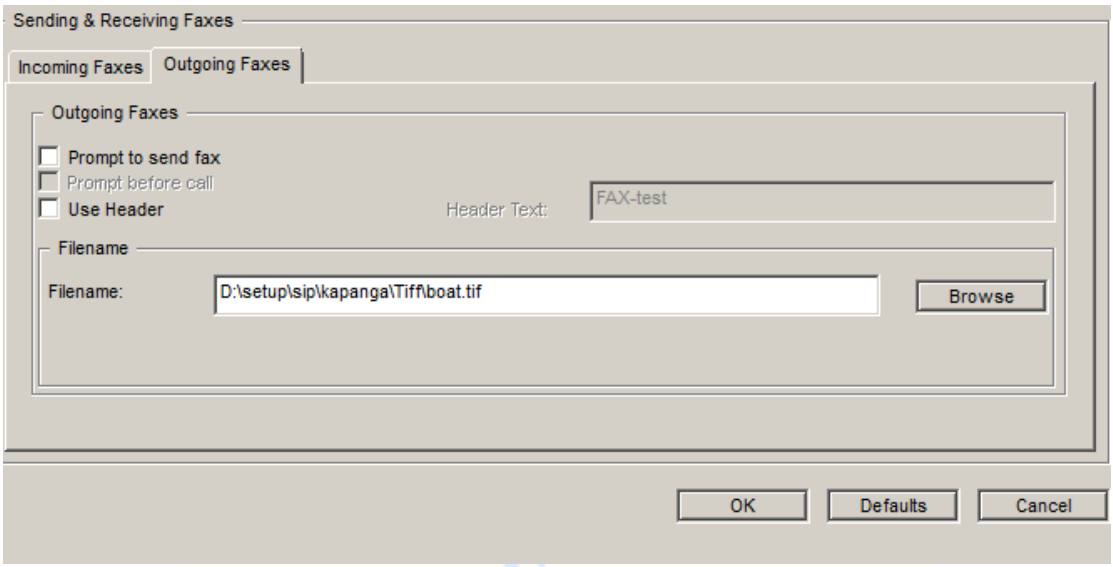
跟原始文件对比，原始文件如下：



发现其中的

Kapanga Softphone [FAX] P.1 07-15-2008 17:29:06

是 kapanga 软件加上了。（偶测试的笔记本的机器时间是 2008 年的）
假如不想要这个头可以在 kapanga 的传真设置里面去掉：



另外发现文件有点变形（船变宽了。呵呵），原因未明

由于收下来的文件变形，实际使用中可能会带来投诉。

(本人之前使用 dialogic 的 hmp 收传真就发生过由于文件变形 被投诉的情况。)

上面说过 kapanga 默认是发送 pdf 文件的，pdf 文件需要安装 acrobat 的 pdf 阅读器才能发送。默认发送的 pdf 文件是 pdf 目录下的 fax.pdf 总共 3 页，FreeSwitch 也是支持收取多页的传真。

fax.pdf 文件 FreeSwitch 收到之后会显示每页的信息：

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:443 ==== Page Received ==========

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:444 Page no = 1

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (
bi-level in the file)

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 238
1 pixels (1728 x 2381 pixels in the file)

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031
/m x 7700/m (8031/m x 7700/m in the file)

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 1-D (
1)

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 208519 bytes

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 2

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 2

2013-07-19 10:34:32.884400 [DEBUG] mod_spandsp_fax.c:452 ==========

2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:443 ==== Page Received ==========

2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:444 Page no = 2

2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (
bi-level in the file)

2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 238
1 pixels (1728 x 2381 pixels in the file)

2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031
/m x 7700/m (8031/m x 7700/m in the file)
2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 1-D (
1)
2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 208951 bytes
2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 2
2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 2
2013-07-19 10:36:31.844400 [DEBUG] mod_spandsp_fax.c:452
==========

2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:443 ==== Page Received
==========

2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:444 Page no = 3
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (
bi-level in the file)
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 238
1 pixels (1728 x 2381 pixels in the file)
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031
/m x 7700/m (8031/m x 7700/m in the file)
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 1-D (
1)
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 209455 bytes
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 2
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 2
2013-07-19 10:38:31.064400 [DEBUG] mod_spandsp_fax.c:452
==========

注意上面日志里面的斜体部分，表示收到 3 页。

163. FreeSwitch 如何实现收传真机发过来的传真？

先准备测试传真一台，
由于要实现落地，需要 VOIP 网关一个。
本人使用迅时的 mx8 网关测试，
接上模拟电话线。在基本配置--系统　上配置支持传真业务和数据业务。否则会不能收发传真。

其它的配置请参考前面的 mx8 配置问题。

拨号计划基本一样，只不过 被叫号码不一样：

```
<extension name="fax_receive">
        <condition field="destination_number" expression="^(12396\d+)$">
            <action application="answer" />
           <action application="set" data="fax_enable_t38=true"/>
           <action application="set" data="fax_enable_t38_request=true"/>
            <action application="playback" data="silence_stream://2000"/>
            <action application="rxfax" data="d:/rxfax-8.tif"/>
            <action application="hangup"/>
        </condition>
    </extension>
```

然后使用传真机拨打。

接通之后可以听见传真音，然后在传真机按发送文件。

FreeSwitch 接收成之后提示如下：

```
2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:443 ==== Page Received
==========================================================
2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:444 Page no = 1
2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (bi-level in the file)
2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 1149 pixels (1728 x 1149 pixels in the file
)
```

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031/m x 3850/m (8031/m x 3850/m in the file)

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 2-D (2)

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 25914 bytes

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 0

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 0

2013-07-17 14:14:00.472101 [DEBUG] mod_spandsp_fax.c:452 ==========

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:511 ==========

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:517 Fax successfully received.

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:529 Remote station id:

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:530 Local station id: SpanDSP Fax Ident

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:531 Pages transferred: 1

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:533 Total fax pages: 1

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:534 Image resolution: 8031x3850

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:535 Transfer Rate: 9600

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:537 ECM status off

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:538 remote country:

2013-07-17 14:14:04.731344 [DEBUG] mod_spandsp_fax.c:539 remote vendor:

…

164. FreeSwitch 如何实现发传真给普通传真机？

发传真通常的定义是系统 作为主叫拨出到用户，然后系统发传真，用户收传真。比如 FreeSwitch 拨出到普通传真机上，然后 FreeSwitch 上发送，传真机上接收。本人测试还是通过迅时的 mx8 voip 网关进行落点。

通过 mx8 发给传真机收。Mx8 的配置同上面。

接上电话线之后，在 FreeSwitch 控制台 或者 FreeSwitch cli 界面上执行：

originate sofia/gateway/gw1/8261568 &txfax(d:/rxfax0.tif)

表示给 8261568 号码发传真

originate sofia/gateway/gw1/2518800 &txfax(d:/rxfax0.tif)

表示给 2518800 号码发传真

测试两个号码都可以收到传真。

发送过程照样有 T.38 的握手,握手消息如下：

2013-07-17 14:22:25.711999 [DEBUG] sofia.c:5696 Remote SDP:

v=0

o=- 17129 481009 IN IP4 192.168.1.251

s=-

c=IN IP4 192.168.1.251

t=0 0

m=image 10010 udptl t38

c=IN IP4 192.168.1.251

a=T38FaxVersion:0

a=T38MaxBitRate:9600
a=T38FaxRateManagement:transferredTCF
a=T38FaxMaxBuffer:200
a=T38FaxMaxDatagram:72
a=T38FaxUdpEC:t38UDPRedundancy

2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1029 T38 SDP Origin = -
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1030 T38FaxVersion = 0
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1031 T38MaxBitRate = 9600
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1032 T38FaxFillBitRemoval = 0
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1033 T38FaxTranscodingMMR = 0
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1034 T38FaxTranscodingJBIG = 0
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1035 T38FaxRateManagement = 'transferredTCF'
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1036 T38FaxMaxBuffer = 200
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1037 T38FaxMaxDatagram = 72
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1038 T38FaxUdpEC = 't38UDPRedundancy'
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1039 T38VendorInfo = ''
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1040 ip = '192.168.1.251'
2013-07-17 14:22:25.732000 [DEBUG] mod_spandsp_fax.c:1042 port = 10010
2013-07-17 14:22:25.732000 [DEBUG] mod_sofia.c:1495 Remote address:port [192.168.1.251:10010] has not changed.
2013-07-17 14:22:25.732000 [DEBUG] sofia_glue.c:175 sofia/external/2518800 image media sdp:
v=0
o=FreeSwitch 1374026960 1374026962 IN IP4 192.168.1.102
s=FreeSwitch
c=IN IP4 192.168.1.102
t=0 0
m=image 15158 udptl t38
a=T38FaxVersion:0
a=T38MaxBitRate:14400
a=T38FaxRateManagement:transferredTCF
a=T38FaxMaxBuffer:2000
a=T38FaxMaxDatagram:400
a=T38FaxUdpEC:t38UDPRedundancy
...
2013-07-17 14:22:25.732000 [DEBUG] sofia_glue.c:175 sofia/external/2518800 image media sdp:
v=0
o=FreeSwitch 1374026960 1374026963 IN IP4 192.168.1.102
s=FreeSwitch
c=IN IP4 192.168.1.102
t=0 0
m=image 15158 udptl t38
a=T38FaxVersion:0
a=T38MaxBitRate:14400
a=T38FaxRateManagement:transferredTCF

a=T38FaxMaxBuffer:2000

a=T38FaxMaxDatagram:400

a=T38FaxUdpEC:t38UDPRedundancy

。。。

发送 成功之后

FreeSwitch 提示如下：

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:443 ==== Page Sent ==========

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:444 Page no = 1

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:445 Image type = bi-level (bi-level in the file)

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:446 Image size = 1728 x 1695 pixels (1728 x 1663 pixels in the file)

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:447 Image resolution = 8031/m x 7700/m (8031/m x 7716/m in the file)

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:448 Compression = T.4 2-D (2)

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:449 Compressed image size = 53521 bytes

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:450 Bad rows = 0

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:451 Longest bad row run = 0

2013-07-17 14:23:27.211516 [DEBUG] mod_spandsp_fax.c:452 ==========

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:511 ==========

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:515 Fax successfully sent.

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:529 Remote station id:

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:530 Local station id: SpanDSP Fax Ident

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:531 Pages transferred: 1

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:533 Total fax pages: 1

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:534 Image resolution: 8031x7700

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:535 Transfer Rate: 9600

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:537 ECM status off

2013-07-17 14:23:29.791664 [DEBUG] mod_spandsp_fax.c:538 remote country:

。。。

说明：

其中 conf\sip_profiles\external\gw1.xml 内容如下

```
<gateway name="gw1">
    <param name="realm" value="192.168.1.251:5060"/>
    <param name="username" value="5678"/>
    <param name="password" value="1234"/>
    <param name="register" value="false" />
</gateway>
```

其中的 ip 就是 mx8 的 ip。

165. FreeSwitch 如何实现发传真给软电话？

发传真通常的定义是系统 作为主叫拨出到用户，然后系统发传真，用户收传真。比如 FreeSwitch 拨出到 Kapanga，然后 FreeSwitch 上发送，Kapanga 上接收。

估计是本人 RP 不行，测试一直无法成功。

看 FreeSwitch 的日志是 fax 的握手都没有握上。Kapanga 收下的文件只有 1kb 大小。

将来使用其它软电话进行测试。

166. FreeSwitch 如何实现发送非 TIF 格式的文件传真？

假如要发送的文件不是 tif 格式的文件， 比如是 word 之类的文件。

需要先使用格式转换的软件工具或者 sdk 进行转换，然后才能发送。本人使用的是:

http://www.i-enet.com/SmartPrinter.htm

sdk 进行格式转换的。

转换的 c 代码非常简单：

```
CConvertAgent *pConvertEngine = NULL;
pConvertEngine = new CConvertAgent();

 if(pConvertEngine)
  {
     iRet = pConvertEngine->InitAgent("SmartPrinter", 60, "Demo","Demo");
      if(iRet == SM_SUCCESS)
      {
          iRet = pConvertEngine->ConvertDoc("c:\\test.doc","c:\\test.tif"); // doc 转 Tiff
      }
  }

 if(pConvertEngine)
  {
      delete pConvertEngine;
      pConvertEngine = NULL;
  }
```

167. FreeSwitch 如何支持 SRTP（加密 RTP）通话？

有些的应用需要对终端的通话进行加密，防止窃听。这个时候就需要语音加密。

假如要支持语音加密，需要支持 SRTP。

所谓 SRTP，就是安全实时传输协议(Secure Real-time Transport Protocol)，在实时传输协议 RTP(Real-time Transport Protocol)基础上所定义的一个协议，具体参见 RFC 3711。

首先需要找到支持 SRTP 的 IP 话机或者软电话。
SRTP 本人测试 IP 话机使用 亿联的 T20P 话机（大约 300 多一个）.
SRTP 本人测试软电话有两个,其中一个是支持 SRTP 的 ImsDroid 软电话.
另外一个是不支持 SRTP 的 eyebeam，使用来测试 SRTP 和普通 RTP 的互通性。

首先测试 SRTP 之间的互通：
使用 ImsDroid 软电话进行测试。
首先在软电话 A 上的
ImsDroid 上的【options】 菜单里面是【Security】菜单里面
SRTP Mode 设置为必须(Mandatory),
语音编码器设置为只有 alaw
在软电话 B 上设置跟 A 一样。
FreeSwitch 设置如下：
conf\sip_profiles\internal.xml
里面修改：

```
<param name="inbound-zrtp-passthru" value="false"/>
```

拨号计划如下：

```
<extension name="Local_Extension2">
        <condition field="destination_number" expression="^([0-9]\d+)$">
                        <action application="export" data="dialed_extension=$1"/>
                        <action application="answer" />
                        <action application="set" data="call_timeout=30"/>
                          <action application="set" data="record_sample_rate=8000"/>
                          <action application="export" data="RECORD_STEREO=false"/>
                        <action application="set" data="hangup_after_bridge=true"/>
                        <action application="set" data="continue_on_fail=true"/>

                                <action application="set" data="sip_secure_media=true"/>
                                  <action application="export" data="sip_secure_media=true"/>
                    <action application="export" data="sdp_secure_savp_only=true" />
                                <action application="playback" data="d:/data/system/zzy_hello.alaw"/>
                <action application="export" data="execute_on_answer=record_session
z:/record/${strftime(%Y-%m-%d-%H-%M-%S)}_${caller_id_name}_${destination_number}.alaw"/>
                        <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
                         </condition>
      </extension>
```

然后开始测试：

A 通过 FreeSwitch 拨打 B
先听到 zzy_hello.alaw 的语音：欢迎使用 SRTP 加密语音测试系统。
语音播完。转接到 B。B 上应答之后 A 和 B 通话正常。对 a 和 b 的通话录音也正常。
观察分析其过程：
FreeSwitch 上拨入的日志如下，我们只关注 SDP 部分：

2013-07-18 16:47:01.703272 [DEBUG] sofia.c:5696 Remote SDP:

v=0

o=doubango 1983 678901 IN IP4 192.168.1.106

s=-

c=IN IP4 192.168.1.106

t=0 0

m=audio 3756 RTP/SAVP 8 101

a=rtpmap:8 PCMA/8000/1

a=rtpmap:101 telephone-event/8000/1

a=fmtp:101 0-16

a=ptime:20

a=silenceSupp:off - - - -

a=tcap:1 RTP/SAVPF

a=pcfg:1 t=1

a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:DWVxmaijgAtTaaM8o/nxSLkZau0pJr/84ggJq3iR

a=crypto:2 AES_CM_128_HMAC_SHA1_32 inline:iI/YaTuHoK5VaQNjUvYrg1zCFQjs19rO/TbzcUbQ

a=rtcp-mux

a=ssrc:4150037250 cname:ldjWoB60jbyQlR6e

a=ssrc:4150037250 mslabel:6994f7d1-6ce9-4fbd-acfd-84e5131ca2e2

a=ssrc:4150037250 label:Doubango

2013-07-18 16:47:01.703272 [DEBUG] sofia_glue.c:5017 Set Remote Key [1 AES_CM_128_HMAC_SHA1_80 inline:DWVxmaijgAtTaaM8o/nxSLkZau0pJr/84ggJq3iR]

2013-07-18 16:47:01.703272 [DEBUG] sofia_glue.c:3182 Set Local Key [1 AES_CM_128_HMAC_SHA1_80 inline:tIAPNiTnphooDihDZ/SgG2k9EIf5lwBFuH+yJhTt]

2013-07-18 16:47:01.703272 [DEBUG] sofia_glue.c:5176 Audio Codec Compare [PCMA:8:8000:20:64000]/[PCMA:8:8000:20:64000]

2013-07-18 16:47:01.703272 [DEBUG] sofia_glue.c:3119 Set Codec sofia/internal/1031@192.168.1.102 PCMA/8000 20 ms 160 samples 64000 bits

2013-07-18 16:47:01.703272 [DEBUG] switch_core_codec.c:111 sofia/internal/1031@192.168.1.102 Original read codec set to PCMA:8

2013-07-18 16:47:01.703272 [DEBUG] sofia_glue.c:5305 Set 2833 dtmf send/recv payload to 101

2013-07-18 16:47:01.703272 [DEBUG] sofia.c:5929 (sofia/internal/1031@192.168.1.102) State Change CS_NEW -> CS_INIT

....

2013-07-18 16:47:01.703272 [INFO] switch_rtp.c:1815 Activating Secure RTP SEND

2013-07-18 16:47:01.703272 [INFO] switch_rtp.c:1795 Activating Secure RTP RECV

2013-07-18 16:47:01.703272 [DEBUG] switch_core_sqldb.c:2350 Secure Type: srtp:AES_CM_128_HMAC_SHA1_80

2013-07-18 16:47:01.703272 [DEBUG] mod_sofia.c:2842 Ring SDP:

v=0

o=FreeSwitch 1374121453 1374121454 IN IP4 192.168.1.102

s=FreeSwitch

c=IN IP4 192.168.1.102

t=0 0

m=audio 15768 RTP/SAVP 8 101

a=rtpmap:8 PCMA/8000

a=rtpmap:101 telephone-event/8000

a=fmtp:101 0-16

a=silenceSupp:off - - - -

a=ptime:20

a=sendrecv

a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:tIAPNiTnphooDihDZ/SgG2k9EIf5lwBFuH+yJhTt

。。。

注意上面的斜体部分。

转出的日志如下,我们同样只关注 SDP 部分：

。。。

2013-07-18 16:47:03.803392 [DEBUG] sofia_glue.c:3182 Set Local Key [1 AES_CM_128_HMAC_SHA1_32 inline:uyJFHrJjSdXRzJ6mPE92Slu8HPdGpTWNPkPz8x5O]

2013-07-18 16:47:03.803392 [DEBUG] sofia_glue.c:2673 Local SDP:

v=0

o=FreeSwitch 1374121709 1374121710 IN IP4 192.168.1.102

s=FreeSwitch

c=IN IP4 192.168.1.102

t=0 0

m=audio 15514 RTP/SAVP 8 101 13

a=rtpmap:101 telephone-event/8000

a=fmtp:101 0-16

a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:uyJFHrJjSdXRzJ6mPE92Slu8HPdGpTWNPkPz8x5O

a=ptime:20

a=sendrecv

。。。

2013-07-18 16:47:06.003518 [DEBUG] sofia.c:5696 Remote SDP:

v=0

o=doubango 1983 678901 IN IP4 192.168.1.105

s=-

c=IN IP4 192.168.1.105

t=0 0

m=audio 23346 RTP/SAVP 8 101

a=rtpmap:8 PCMA/8000/1

a=rtpmap:101 telephone-event/8000/1

a=fmtp:101 0-16

a=ptime:20

a=minptime:20

a=maxptime:20

a=silenceSupp:off - - - -

a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:ul9GMsQJuuwzgAvTJd8sviL3zJsfonErOrlEzgmt

a=ssrc:4036488365 cname:e3c4371722d69ddc4006b1b797861097

a=ssrc:4036488365 mslabel:6994f7d1-6ce9-4fbd-acfd-84e5131ca2e2

a=ssrc:4036488365 label:doubango@audio

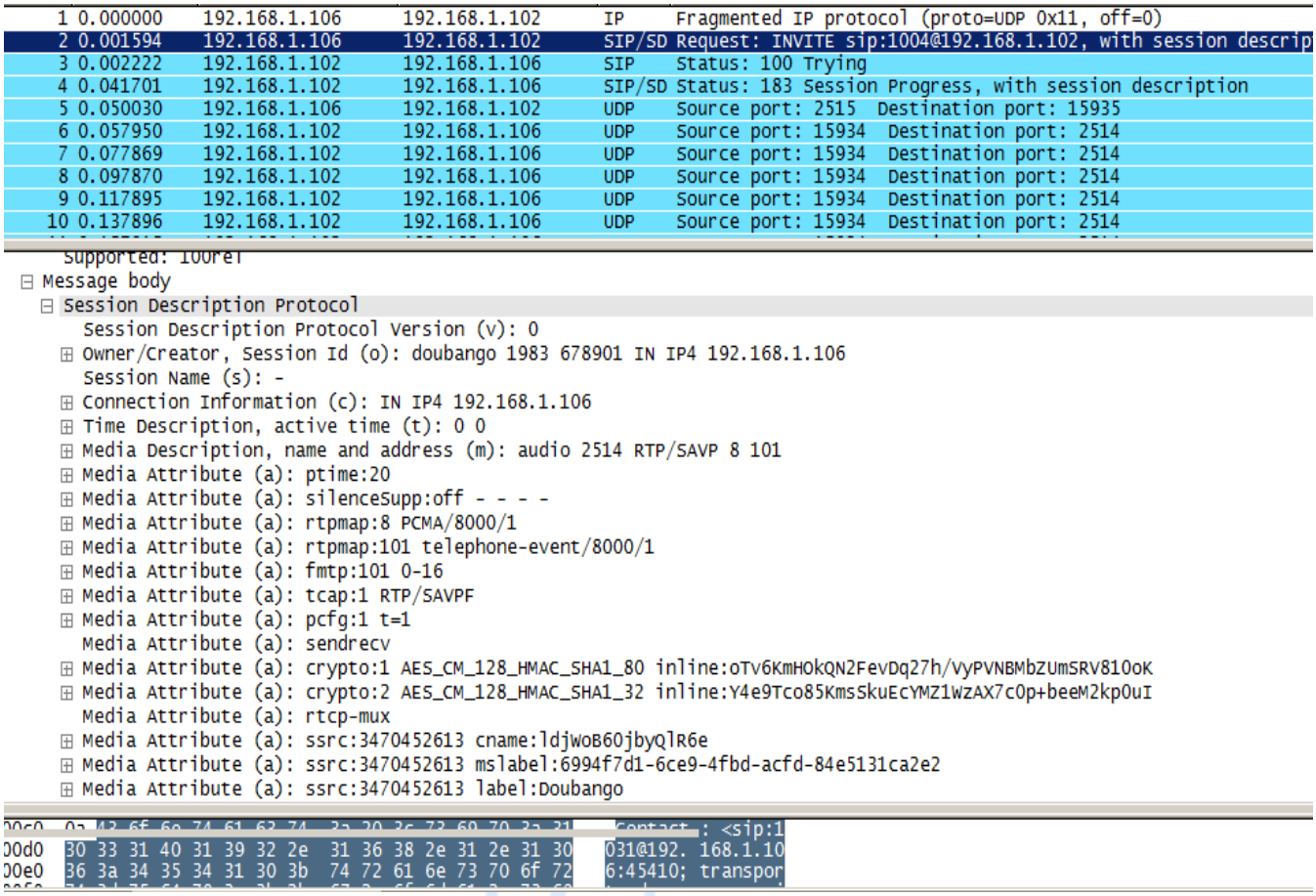
…

2013-07-18 16:47:06.003518 [DEBUG] sofia_glue.c:5017 Set Remote Key [1 AES_CM_128_HMAC_SHA1_32 inline:ul9GMsQJuuwzgAvTJd8sviL3zJsfonErOrlEzgmt]

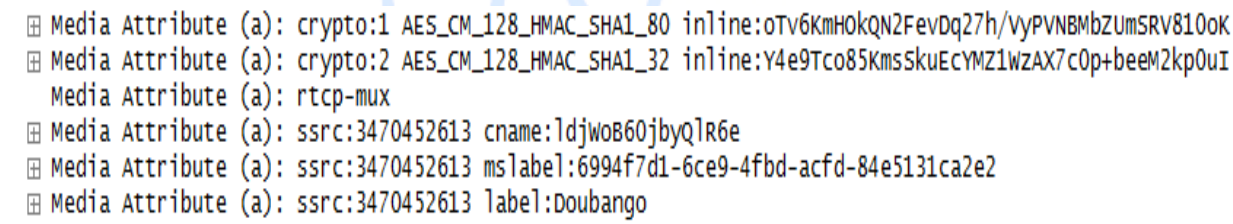
2013-07-18 16:47:06.003518 [DEBUG] sofia_glue.c:5176 Audio Codec Compare [PCMA:8:8000:20:64000]/[PCMA:8:8000:20:64000]

2013-07-18 16:47:06.003518 [DEBUG] sofia_glue.c:3119 Set Codec sofia/internal/sip:1004@192.168.1.105:33326 PCMA/8000 20。。。

在 FreeSwitch 上对 a 的 ip 抓包结果如下：



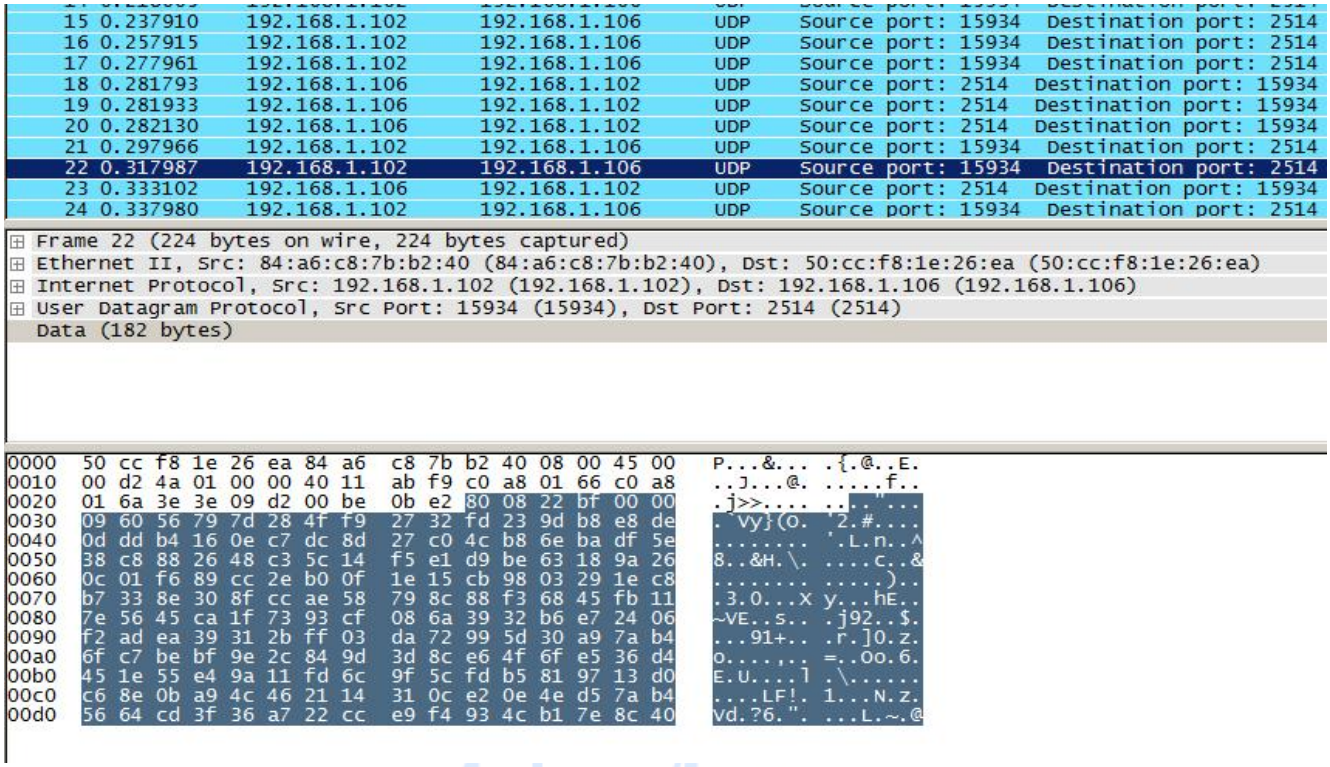
可以看到 sdp 里面有 SRTP 加密的东东：



再看语音的 rtp 包，正常 alaw 是一个包内容 172=12（rtp 头）+160Byte（20ms 的语音）

现在加密之后是 182B。

整个 20ms 的 UDP 包现在是 224B，假如没有加密 是 214B，多出了 10Byte，猜测是加密的开销。



然后测试 SRTP 和 RTP 之间的互通：

A. SRTP 的 ImsDroid 之间是可以通话的，而且 SRTP 的 ImsDroid 拨打到不支持 SRTP 的 eyebeam 上，也可以进行对话。

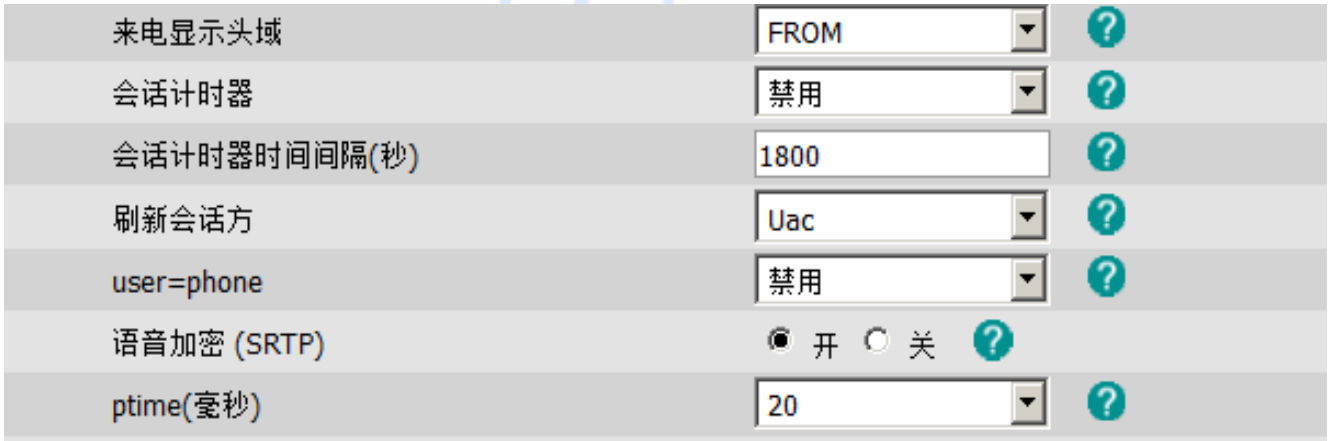
B. 倒过来不支持 SRTP 的 eyebeam 也可以拨打到 SRTP 的 ImsDroid。

最后使用亿联 T20P 进行 SRTP 测试：

首先使用正常没有 SRTP 的方式注册到 FreeSwitch，测试可以了之后。使用默认帐号 admin 和密码 admin 登录到亿联话机的

Http://192.168.1.10 上进行配置。

在【帐号】设置里面的【高级】里面打开 SRTP：

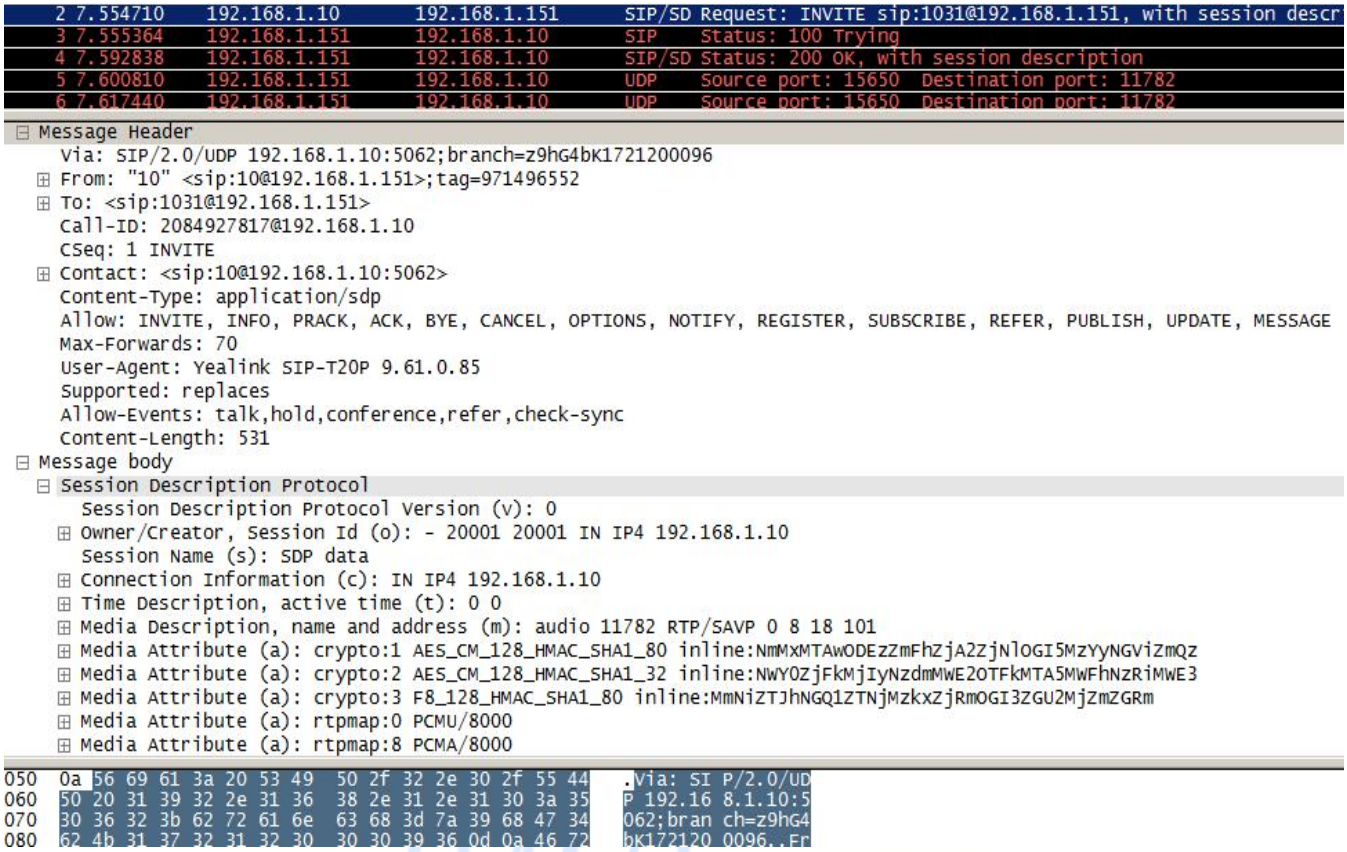


然后跟 ImsDroid 进行测试 SRTP 到 SRTP 的通话，没有问题之后，

再测试跟 eyebeam 的 SRTP 到 RTP 或者 RTP 到 SRTP 之间的通话。

情况跟是 ImsDroid 之间一样的，都可以正常通话，T20P 不管跟 ImsDroid 还是跟 eyebeam 之间都是可以正常通话的。

下面是对 T20P 抓包结果，可以看到支持 3 个密钥：



其它一些说明：

A. 使用 SRTP 之后到 ESL ivr 上 播音录音没有问题，

但是取 2833 的 DTMF 按键是不行的，估计需要使用 sip info 或者 inband 之类的。这个为进行测试，大家有兴趣可以测试。我认为 DTMF 走 sip info 应该是可以的。

B. 根据密钥的不同，有一种是 80bit（10byte 的散列密钥）对带宽来说就是 多出了 10*50（每秒 50 个包）*8=4Kb/s 每线多出了 4K 带宽.另外一种是 32bit（4byte）的散列密钥。

32bit 的密钥发生在 eyebeam 呼叫带 SRTP 的 ImsDroid 的时候，

FreeSwitch 日志如下，我们同样只关注 SDP 部分：

...

2013-07-18 21:31:46.002706 [DEBUG] sofia.c:5696 Remote SDP:

v=0

o=doubango 1983 678901 IN IP4 192.168.1.106

s=-

c=IN IP4 192.168.1.106

t=0 0

m=audio 56474 RTP/SAVP 8 101

a=rtpmap:8 PCMA/8000/1

a=rtpmap:101 telephone-event/8000/1

a=fmtp:101 0-16

a=ptime:20

a=silenceSupp:off - - - -

a=crypto:1 AES_CM_128_HMAC_SHA1_32 inline:m6RdoI+hGaOYdQAq8Cq1wq2Vjc4L+NPCAbBUX5Rm

a=ssrc:3982512707 cname:ldjWoB60jbyQlR6e

a=ssrc:3982512707 mslabel:6994f7d1-6ce9-4fbd-acfd-84e5131ca2e2

a=ssrc:3982512707 label:Doubango ....

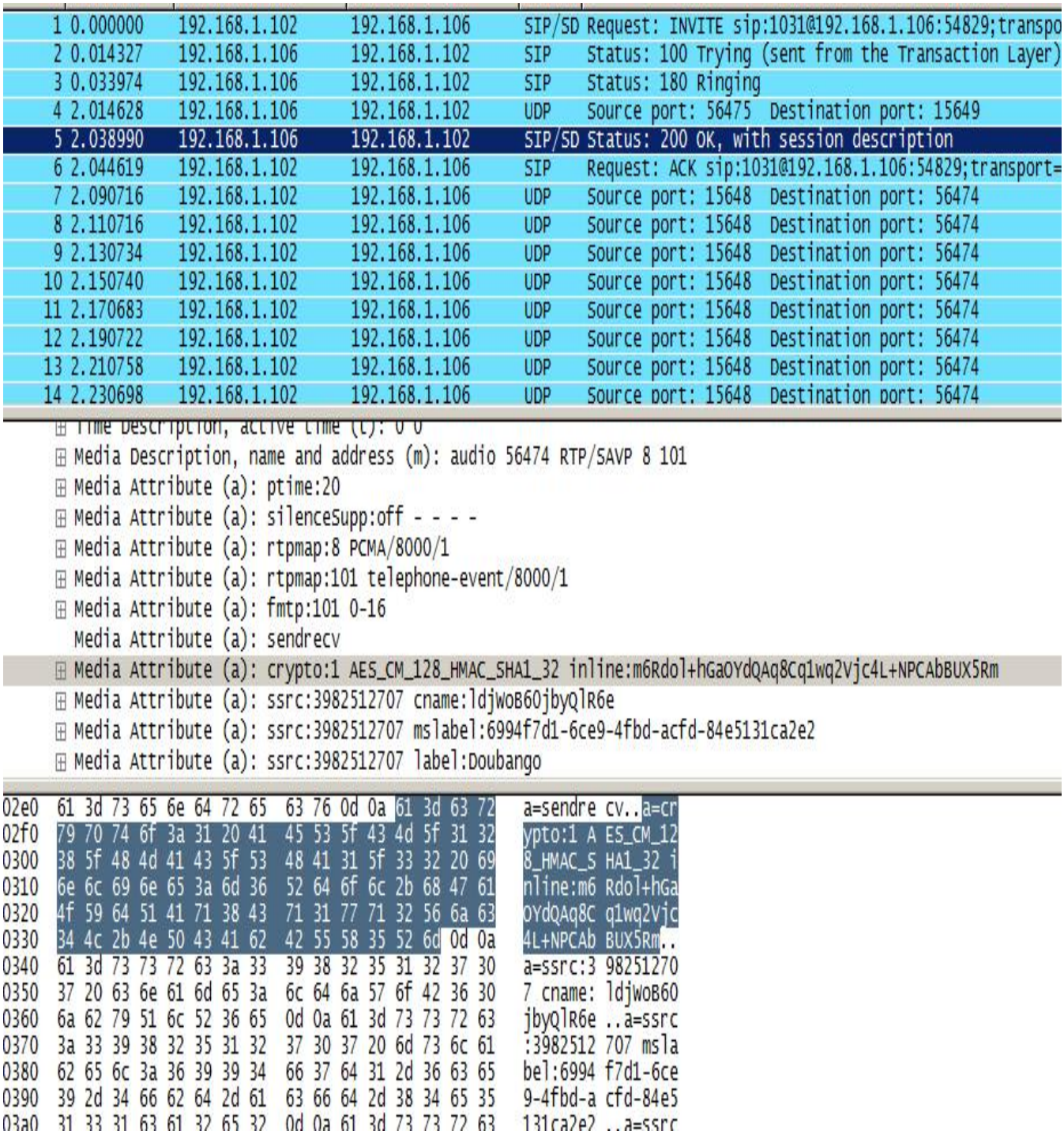
2013-07-18 21:31:46.002706 [INFO] switch_rtp.c:1815 Activating Secure RTP SEND

2013-07-18 21:31:46.002706 [INFO] switch_rtp.c:1795 Activating Secure RTP RECV

2013-07-18 21:31:46.002706 [DEBUG] switch_core_sqldb.c:2350 Secure Type: srtp:AES_CM_128_HMAC_SHA1_32

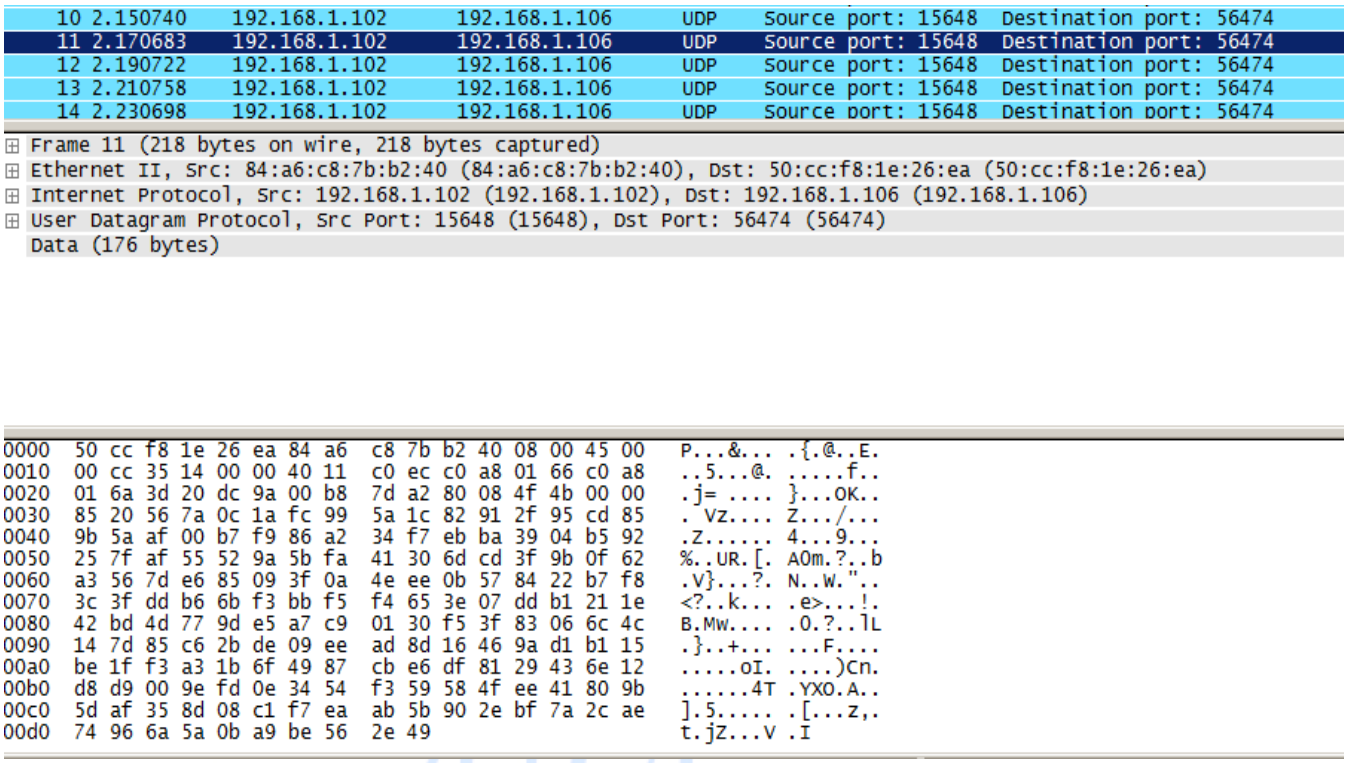
2013-07-18 21:31:46.002706 [DEBUG] switch_core_sqldb.c:2350 Secure Type: srtp:AES_CM_128_HMAC_SHA1_32

... 抓包如下图：



加密只有 4Byte 的开销，这样的话，每线多出的带宽是 4*50（每秒 50 个包）*8=1.6Kb/s 也就是多出了

1.6K 带宽，加密之后的 rtp 如下图，UDP 包是 218Byte，负载是 176，比默认的 172（12Byte 的 RTP 包头+160Byte 的真正语音负载）多了 4Byte。



第十章 FreeSwitch ESL 编程部分

168. FreeSwitch ESL 是什么？

ESL 就是 Event Socket Library，翻译过来就是事件套接字库。

说白了就是其它语言通过这个的事件套接字库编写的脚本或者程序可以控制 FreeSwitch 的行为，比如 呼叫，播音，录音啥的。

169. FreeSwitch ESL 编程能做啥用？

本人的理解假如没有 ESL 编程控制（或者其它的脚本编程控制比如 Erlang），FreeSwitch 仅仅只会是一个 SIP Server。

虽然通过拨号计划可以实现一些功能，但是实际应用的功能往往很难全部使用拨号计划实现。

有了 ESL 编程的控制，FreeSwitch 的功能可以扩展到整个呼叫中心之类的复杂应用。

ESL 编程的控制可以完成最基本的 IVR 系统的全部功能，包括但不限于如下 10 个 IVR 基本功能内容：

1） 接受用户拨入，应答呼叫
2） 系统拨出到用户，
3） 挂断用户呼叫
4） 连接用户 a 和 b 进行媒体通话
5） 断开用户 a 和 b 的媒体通话，但是用户 a 和 b 的信令都没有断开
6） 播音（同时可以获取用户按键）
7） 录音（同时可以获取用户按键）
8） 参加会议
9） 会场控制，设置某个人静音或者允许某个人说话
10） 会场录音的开启和停止。

下面的 ESL 编程控制内容就是针对上面这些基本的 IVR 应用进行描述的。

由于 ESL 原生使用 C/C++语言，本人使用 C/C++进行 ESL 的 IVR 编写。

使用 C/C++的好处是当 ESL 有 bug 崩溃的时候可以直接 debug，从而定位问题。

170. FreeSwitch ESL LIB 例子有哪些？

API 开发 demo

source ：libs/esl 目录下
testserver.c 外联模式例子
testclient.c 内联模式例子
FS_Cli.c FreeSwitch client 的 code

171. FreeSwitch ESL LIB 例子 windows 下如何方便建立 VS 工程？

以 testserver.c 为例子建立 VS 工程
Windows 的 ESL LIB 例子 demo ,有一个已经存在的工程：
\libs\esl 目录下 FS_Cli.2010.vcxproj
我们可以在这个工程的基础上很快建立 例子的工程

使用 Vs2010 打开这个过程
然后把 testserver.c
文件里面的内容拷贝覆盖 FS_Cli.c 里面的内容
然后编译，保存就可以了。

172. FreeSwitch ESL LIB 例子 windows 下使用有哪些要注意？

Windows 下使用 Socket 的程序在使用 Socket 之前必须调用 WSAStartup 函数
以 testserver.c 例子为例：在 main()最开始的地方的
初始化前要增加下面这些代码：

```
WSADATA Data;
  if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
  {
        return false;
  }
```

否则 testserver.c 启动会马上退出。
如下面：

```
int main(void)
{
  WSADATA Data;
  if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
  {
        return 0;
  }

  esl_global_set_default_logger(7);
  esl_listen_threaded("localhost", 8084, mycallback, 100000);

  return 0;
}
```

mycallback 的函数，每个呼叫会多一个线程。
因此 mycallback 里面是多线程的。

其它说明：
除非特别说明，以后的函数代码除了内联模式的外拨线程之外
全部都是在 mycallback 函数里面调用执行的。

173. FreeSwitch 什么时候需要用到内联模式编程？

内联模式在有 ivr 系统外拨（originate）到分机或者其它电话的时候用到。
内联模式初始化如下：

```
int main(void)
{
  WSADATA Data;
  if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
  {
        return 0;
  }
  esl_global_set_default_logger(6);
  int res=esl_connect(&ghandle, "127.0.0.1", 8021, NULL, "ClueCon");
  if(res!=0)
  {
        disp_msg("System Start esl_connect res=%d fail",res);
        return 0;
  }
 。。。。。
```

ghandle 是全局的 ，将来通过 ghandle 可以 产生 uuid，从而进行外拨呼叫

174. FreeSwitch ESL 外联和内联模式编程有啥区别？

ESL 的内联 和外联模式 会让人迷糊，很多人跟电话的拨入（inbound） 拨出（outbound）混淆了。实际上不是一样的概念，一个是电话的呼叫方向，一个是编程程序控制模式。

内联模式在有 ivr 系统外拨（originate）到分机或者其它电话的时候用到。

内联模式 inbound 一开始可以连接好，可以用来下命令。比如 外拨命令 originate。inbound 是直接接受外部指令执行， FS_Cli 就是个典型的例子。

外联模式 outbound 是拨号计划里面电话来电的时候连接到应用的。主要是标识来电 的线程启动。

通常的 ivr 应用，不需要转出或者自动拨出的。outbound 就可以实现了。

175. FreeSwitch 系统拨出如何设置显示的主叫用户名称和主叫号码？

在 originate 的时候可以指定 origination_caller_id_number 参数来在被叫上显示主叫号码，可以使用 origination_caller_id_name 参数来指定在被叫上显示主叫用户名称，

比如：使用 ESL 呼叫 1002 分机的时候要在 1002 分机上显示来电的号码和名称：

originate {originate_timeout=30,origination_caller_id_number=0123456,
origination_caller_id_name=test}sofia/internal/1002@192.168.1.101 0123456

176. FreeSwitch 内联模式如何实现 IVR 全业务外拨用户功能？

FreeSwitch 的 bridge 是连接有一个用户 桥接另外一个用户。不能用来实现 ivr 的外拨功能。比如外拨一个用户让播放提示语音，让它参加会议。

要实现一个 ivr 全业务的外拨功能意味着：外拨接通之后的被叫用户可以进入 ivr，可以和其它用户通话，或者参加会议。还要求外拨以后可以随时挂掉正在外拨的呼叫。

要实现以上功能需要如下步骤：

1）首先，使用下面的函数获取一个新的 UUID

```
int get_uuid(char*uuid)
{
        uuid[0]=0;
        esl_send_recv(&ghandle, "api create_uuid\n\n");
        if (ghandle.last_sr_event && ghandle.last_sr_event->body)
        {
                disp_msg("get_uuid:[%s]\n", ghandle.last_sr_event->body);
                strcpy(uuid,ghandle.last_sr_event->body);
                return 0;
        }
        return 1;
}
```

ghandle 是全局的

这个 uuid 将来用来外拨，或者用来挂掉正在外拨的呼叫

2）创建一个专门负责呼叫的线程：

```
CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)CallOutWorkThreadRealTime,(LPVOID)(index),0, NULL);
```

该线程新建立一个内联模式的连接到 FreeSwitch，然后 根据被叫号码的长度判断是内部分机呼叫 还是外拨到网关。

主要代码如下：

```
int CallOutWorkThreadRealTime(void* arg)
 {
。。。。
 res=esl_connect(&handle, "127.0.0.1", 8021, NULL, "ClueCon");
 if(res!=0)
 {
        disp_msg("esl_connect esl_connect res=%d fail",res);
        return 0;
```

```
}

if(strlen(called)>4)
        sprintf(cmd_tmp,"bgapi originate {origination_uuid=%s,originate_timeout=60}sofia/gateway/gw1/%s %s\n\n",uuid,called,system_caller);
else
        sprintf(cmd_tmp,"bgapi originate {origination_uuid=%s,originate_timeout=30}sofia/internal/%s%c%s %s\n\n",uuid,called,'%',FreeSwitchserverip,system_caller);
res=esl_send_recv(&handle,cmd_tmp);
disp_msg("CallOutThread esl_send_recv:[%s],res=%d",cmd_tmp,res);

if (handle.last_sr_event && handle.last_sr_event->body)
{
        disp_msg("[%s],index=%d", handle.last_sr_event->body,index);
}
else
{
        disp_msg("[%s],index=%d,last_sr_reply", handle.last_sr_reply,index);
}
disp_msg("CallOutThread uuid=[%s]",uuid);
esl_filter(&handle, "unique-id", uuid);
esl_events(&handle, ESL_EVENT_TYPE_PLAIN, "SESSION_HEARTBEAT CHANNEL_ANSWER CHANNEL_ORIGINATE CHANNEL_PROGRESS CHANNEL_HANGUP "
                "CHANNEL_BRIDGE CHANNEL_UNBRIDGE CHANNEL_OUTGOING CHANNEL_EXECUTE CHANNEL_EXECUTE_COMPLETE DTMF CUSTOM conference::maintenance");

esl_send_recv(&handle, "linger");
。。。。。
```

说明：originate_timeout=60 表示外拨超时时间，被叫超过这个时间没有应答，ivr 系统就收到 CHANNEL_HANGUP。

3）然后使用 check_makecall_event()和 check_makecall_body()函数监听 下面 3 个事件：

"CHANNEL_HANGUP" 用户没有接，挂机了

"CHANNEL_PROGRESS" 用户开始振铃，这个事件不是一定有的，假如设置自动应答，可能就没有

"CHANNEL_ANSWER"用户开始应答

获取这几个事件是创建一个专门负责呼叫的线程的最核心的功能，假如不需要获取上面的 3 个事件，就不需要创建一个专门负责呼叫的线程，可以直接使用全局的 ghandle 来执行“bgapi originate...” 实现外拨，被叫用户接通之后也会到外联的 callback 函数里面。但是完备的 ivr 系统，在外拨的过程中，需要获取上面的这些事件进行控制。

check_makecall_event()和 check_makecall_body()函数如下：

```
int check_makecall_body(const char*eventbody,char*dtmfbuf,int maxdtmf,char* endchar,int *channel_execute)
{
```

```
 char tmp[128];
 unsigned int i,len;

 if (eventbody==NULL) return 0;
 len=strlen(eventbody);
 if(len>64) len=64;
 strncpy(tmp,eventbody+strlen("Event-Name: "),len);
 tmp[len]=0;
 for(i=0;i<strlen(tmp);i++) if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
 disp_msg("Event-Name:[%s]", tmp);
 if(strcmp(tmp,"CHANNEL_EXECUTE")==0) *channel_execute=1;
 if(strcmp(tmp,"CHANNEL_HANGUP")==0) return -98;
 if(strcmp(tmp,"CHANNEL_PROGRESS")==0) return EV_ALERTING;
 if(strcmp(tmp,"CHANNEL_ANSWER")==0)      return EV_CONNECTED;
 return 0;
}
int check_makecall_event(esl_handle_t *handle,int timer)
{
 int done = 0,c=0;
 esl_status_t status;
 time_t exp = 0;
 char dtmf[128];
 int res=0;
 int    channel_execute=0;
 dtmf[0]=0;
 time_t time1=time(NULL);

 while((status = esl_recv_timed(handle, 1000)) != ESL_FAIL)
 {
        c++;
        if(timer>0 && (time(NULL)-time1>=timer)){res=100;break;}
        if (status == ESL_SUCCESS)
        {
                const char *type = esl_event_get_header(handle->last_event, "content-type");

                if (type)
                {
                        if(strcasecmp(type, "text/disconnect-notice")==0)
                        {
                                const char *dispo = esl_event_get_header(handle->last_event, "content-disposition");
                                disp_msg("Got a disconnection notice dispostion: [%s]", dispo ? dispo : "");
                                if (dispo && strcmp(dispo, "linger")==0)
                                {
                                        res=-99;
```

```
                        break;
                    }
                }
                if(strcasecmp(type, "text/event-plain")==0)
                {
                    const char *eventbody=esl_event_get_body(handle->last_event);
                    if((res=check_makecall_body(eventbody,dtmf,1,"#",&channel_execute)))
                    {
                        disp_msg("check_makecall_body res=%d.",res);
                        if(res)     break;
                    }
                }
            }
        }
    }
    disp_msg("check_makecall_event res=%d",res);
    return res;
}
```

4）最后，特别说明：

假如用户挂机本线程结束。

假如用户接通本线程也结束。

假如用户接通，FreeSwitch 将自动马上呼叫 system_caller，通过外联模式的 callback 进入到处理的 ivr 函数里面。

这样就统一了拨入和拨出的 ivr 处理，将拨出当作拨入一起处理，处理 ivr 的全业务。（为啥这样做，是因为一来这样统一处理比较方便，二来本人测试，直接在外拨的线程里面处理 ivr 的业务，会有莫名其妙的情况发生。）

在外联模式 callback 函数中，可以通过下面的办法来知道是 用户拨入还是系统拨出：

```
int dir=1;//call in
 if(esl_event_get_header(handle.info_event, "call-direction"))
{
 disp_msg("dir:%s",esl_event_get_header(handle.info_event, "call-direction"));
 if(strcmp(esl_event_get_header(handle.info_event,"call-direction"),"outbound")==0) dir=0;
 }
```

177. FreeSwitch 如何设置支持其它机器内联模式到 FreeSwitch?

修改 conf\autoload_configs\event_socket.conf.xml 配置文件：

将里面的 <param name="listen-ip" value="127.0.0.1"/> 改为

<param name="listen-ip" value="0.0.0.0"/>

这样其它机器上的 esl 客户端可以内联到 FreeSwitch 的 ip 上。

而不仅仅是本机的 127.0.0.1 的 ip。

另外将来的代码 esl_connect(&ghandle, FreeSwitchip, 8021, NULL, "ClueCon");

FreeSwitchip 不能是127.0.0.1 因为是FreeSwitch的网卡上的ip。

178. FreeSwitch 如何设置支持 SOCKET EVENT API 外联模式编程？

在 \conf\dialplan\default.xml 配置文件里面加上：

```
<extension name="socket">
        <condition field="destination_number" expression="^12396$">
        <action application="socket" data="127.0.0.1:8084 async full"/>
        </condition>
    </extension>
```

其中的 12396 表示接入号码。也就是上面内联模式里面的 system_caller。
之后，拨打 12396 的号码就会到本地的 8084 端口的服务流程（用户自己控制的 ivr）上
说明：
假如只有用户拨入到 ivr 系统，没有 ivr 系统拨出到用户的情况，使用外联模式一种就足够了，不需要使用 内联模式。

179. FreeSwitch 如何支持外联到其它机器的 ESL 客户端？

修改上面拨号计划里面的

```
<action application="socket" data="127.0.0.1:8084 async full"/>
```

类似 内线模式的 ip，上面的 127.0.0.1 ip 可以设置为其它机器的 ip，那么 ESL 的客户端就可以运行在其它机器上。当有电路来的时候，FreeSwitch 会外联到这个指定的 ip 上。
另外在代码里面esl 的客户端侦听的ip要设置为如下：

```
esl_listen_threaded("0.0.0.0", 8084, mycallback, 100000);
```

180. FreeSwitch ESL 外联模式同步和异步模式有啥区别？

A. 设置区别：
在 \conf\dialplan\default.xml 配置文件里的<extension name="socket"> 章节里面：

异步模式参数：

```
<action application="socket" data="127.0.0.1:8084 async full"/>
```

同步模式参数：

```
<action application="socket" data="127.0.0.1:8084 sync full"/>
```

B. 使用区别：主要区别在于 执行 app 或者 api 的时候异步模式是马上返回的。
而同步模式是阻塞的。一直到条件满足函数才会返回的。
C. 例子： 以 播音举例

```
esl_execute(handle, "play_and_get_digits", "1  12  1  15000  #  intelno.wav" , NULL);
```

其中的"1 12 1 10000 # intelno.wav" 分别对应下面参数：
<min> <max> <tries> <timeout ms> <terminators> <file>

表示 最小收 1 个按键 最大收 12 个按键 重复 1 次 等 15000 毫秒，按#结束。提示语音是 intelno.wav。

异步模式下 esl_execute 执行之后马上返回，ivr 可以进入处理收按键环节。

用户开始听见 intelno.wav 的语音内容。

在异步模式下，用户的按键处理条件可以是程序自己来处理。按键的结束条件也不仅仅只有上面几个，比如可以设置 两个按键的间隔时间等参数。这样大大增加了灵活性。

同步模式下，函数 esl_execute 执行就没有返回，除非以下几个条件之一达到：

用户按键 达到 12 个，

用户按了#键

语音播放结束 超过 15 秒时间。

函数才会返回。

表面上看同步模式比较简单，开发流程比较容易。

但是一旦是多线并发，业务复杂的流程，比如使用 esl 开发呼叫中心的流程，或者其它的复杂的流程。同步模式处理起来就吃力。简单就意味着不够灵活。

为了系统将来的业务扩展支持，我们建议使用异步模式。

以下的例子都是以异步模式作为说明。

181. FreeSwitch ESL 开发如何理解 IVR 的播音取按键条件？

Ivr 系统最常用的是播音，然后取按键 的动作。

典型是情况是播语音菜单，或者播提示输入一串号码的语音提示。

该动作 的结束条件很多，初学者往往稀里糊涂的。

讨论如下：

A. ivr 系统先播音，过程可以按键打断，进入收按键后续动作这个 FreeSwitch 本身就是支持的

B. 播音正常结束。等待用户输入按键

C. 进入收用户输入按键动作之后：有很多条件是并发的

这个些条件通常可以分为：

最大允许输入几个按键

按某个或者某些特殊按键，比如*或者#结束输入

全部按键的最大允许时间，

两个按键之间的允许最大间隔时间

典型的情况是一旦上面的 4 个条件任何一个达到条件，就要完成整个播音取按键动作。上面的这 4 种情况能满足大多数的 ivr 业务需要。

。。。当然还可以自己设计条件，比如包括播音的整个动作的允许时间，但是很少的情况会使用到这些额外的。

182. FreeSwitch ESL 开发如何支持连续播多个语音获取按键？

以异步模式为例子：

主要由以下几个函数组成，说明如下：

//播音取按键

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

//参数:handle:会话 handle

//uuid： 会话的 id

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//EndDtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//dtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//说明:假如只有播音,不收取按键 设置：MaxDtmf=0

//member_id 会议的成员 id >0 =0 表示没有进入会议

```
int play_get_dtmf(esl_handle_t *handle,char*uuid,char*filename0,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int &member_id)
{
 int res=0;
 char *p1,*p2,*p,*pan;
 char files[MAXFILES][100];
 int count,i,any_stop=0;
 char tmp[1024],cmd_tmp[1024],enddtmf[128],outdtmfbuff[128],filename[128],filename1[128];
 if(EndDtmf==NULL || EndDtmf[0]==0)
        strcpy(enddtmf,"q");
 else
        strcpy(enddtmf,EndDtmf);
 if(outdtmf) outdtmf[0]=0;
 p1=filename0;
 p2=p1;
 count=0;
 while ((*p2)!='\0')
 {
        p2++;
        if( *p2!=';') continue;
        strcpy(tmp,p1);
        tmp[p2-p1]='\0';
        strcpy(files[count++],tmp);
        p2++;
        if(*p2==';') p2++;
        p1=p2;
        if(count>=MAXFILES) break;
 }
 if(p2>p1)
 {
```

```
        strcpy(files[count++],p1);
        files[count][p2-p1]='\0';
    }
    if(count<=0 || count>=MAXFILES)
    {
        disp_msg("count<2 || count>=MAXFILES _file=[%s]",filename0);
        return -1;
    }
    for(i=0;i<count;i++)
    {
        pan=strstr(files[i],":");
        p=strstr(files[i],"\\");
        if(p==NULL) p=strstr(files[i],"/");
        if(p==NULL)
        {
            sprintf(filename1,"%s/%s",_subdir,files[i]);
        }
        else
            strcpy(filename1,files[i]);

        if(pan==NULL)
        {
            if(strstr(filename1,".")==0)
                sprintf(filename,"%s/%s.alaw",playdir,filename1);
            else
                sprintf(filename,"%s/%s",playdir,filename1);
        }
        else
        {
            if(strstr(filename1,".")==0)
                sprintf(filename,"%s.alaw",filename1);
            else
                sprintf(filename,"%s",filename1);
        }
        if(_access(filename,0)==-1)
        {
            strcpy(tmp,filename);
            strtok(tmp,".");
            strcat(tmp,".pcm");
            if(_access(tmp,0)==-1)
                sprintf(filename,"%s/%s/default.alaw",playdir,_subdir);
            else
                copyfile(tmp,filename);//pcm==>alaw.
        }
```

```
	disp_msg("play_get_dtmf %s,MaxDtmf=%d,MaxTimer=%d,TwoDtmfTimer=%d,EndDtmf=[%s]",filename,MaxDtmf,MaxTimer
,TwoDtmfTimer,EndDtmf);
		//<min> <max> <tries> <timeout ms> <terminators> <file>
		sprintf(cmd_tmp,"1 1 1 10 a %s",filename);//设置让 play_get_dtmf 函数里面的按键处理条件无效，ivr
check_dtmf_event 函数自己处理按键判断
		esl_execute(handle, "play_and_get_digits",cmd_tmp, NULL);
		memset(outdtmfbuff,0,sizeof(outdtmfbuff));
		if(i<count-1)
	res=check_play_dtmf_event_nowaitdtmf(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtmfTimer,
	outdtmfbuff,member_id);
		else		res=check_play_dtmf_event(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtmfTimer,outdtmfbuff,
	member_id);
		if(outdtmf) strcat(outdtmf,outdtmfbuff);
		disp_msg("play_get_dtmf %s end,dtmf=[%s],res=%d",filename,outdtmfbuff,res);
		if(res!=1)
		{
			break;
		}
	}
	if(outdtmf) disp_msg("play_get_dtmf %s all end,dtmf=[%s]",filename,outdtmf);
	return res;
}
```

//播音之后的取按键检查函数

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

//参数:handle:会话 handle

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//enddtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//其它说明:假如只有播音,不收取按键 设置：MaxDtmf=0

//member_id 会议的成员 id >0 =0 表示没有进入会议

int check_play_dtmf_event(esl_handle_t *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int &member_id)

//播音之后的取按键检查(不等待)函数

//当多个语音的时候，不是最后一个语音播放的时候使用这个函数。播放完毕就返回，不等待按键，继续播放下一个语音

int check_play_dtmf_event_nowaitdtmf(esl_handle_t *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int

TwoDtmfTimer,char*outdtmf,int&member_id)

这个函数和 check_play_dtmf_event（）区别在于一个要等等，一个不要等，实现上基本类似。

//播音录音的事件体内容检查函数：

```
int check_play_record_body(esl_handle_t *handle,char*uuid,const char*eventbody,char*dtmfbuf,int maxdtmf,char*
endchar,int *channel_execute,int &member_id)
{
    char tmp[128],cmd_tmp[1024],*p;
    unsigned int i,len;

    if (eventbody==NULL) return 0;
    len=strlen(eventbody);
    if(len>64) len=64;
    strncpy(tmp,eventbody+strlen("Event-Name: "),len);
    tmp[len]=0;
    for(i=0;i<strlen(tmp);i++) if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
    if(strcmp(tmp,"CHANNEL_EXECUTE_COMPLETE")==0)
    {
        if(*channel_execute){ disp_msg("Event-Name:[%s]", tmp);return 1;}
        disp_msg("Event-Name:[CHANNEL_EXECUTE_COMPLETE] invalid");
        return 0;
    }

    disp_msg("Event-Name:[%s]", tmp);
    if(strcmp(tmp,"CHANNEL_EXECUTE")==0) *channel_execute=1;
    if(strcmp(tmp,"CHANNEL_HANGUP")==0) return -98;
    if(tmp[0]='s' && tmp[1]=='s')//strcmp(tmp,"ss: conference%3A%3Amaintenance")==0)
    {
        if(member_id==0)
        {
            p=(char*)strstr(eventbody,"Member-ID: ");
            if(p!=NULL)
            {
                strcpy(tmp,p+strlen("Member-ID: "));
                for(i=0;i<strlen(tmp);i++)if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
                member_id=atoi(tmp);
                disp_msg("member_id:%d", member_id);
            }
        }
        return 0;
    }
    if(_strnicmp(eventbody,"Event-Name: DTMF",strlen("Event-Name: DTMF"))==0)
    {
        char*p;
```

```
            p=(char*)strstr(eventbody,"DTMF-Digit: ");
            if(p!=NULL)
            {
                *channel_execute=0;
                strcpy(tmp,p+strlen("DTMF-Digit: "));
                for(i=0;i<strlen(tmp);i++)if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
                if(strcmp(tmp,"%23")==0) strcpy(tmp,"#");
                disp_msg("dtmf:[%s]", tmp);
                if(dtmfbuf && strlen(dtmfbuf)<64) strcat(dtmfbuf,tmp);
                if(strlen(dtmfbuf)>=maxdtmf) return 2;
                len=strlen(endchar);
                //支持最大 3 个结束按键
                if(len>0) if(tmp[0]==endchar[0]) return 3;
                if(len>1) if(tmp[0]==endchar[1]) return 3;
                if(len>2) if(tmp[0]==endchar[2]) return 3;
            }
        }
        return 0;
    }
```

使用例子如下 ：

```
//播音，语音播放结束不等按键，
play_get_dtmf(&handle,"hello1;hello2","#",0,0,0,dtmf, member_id);
//播音同时取按键，语音播放结束等用户按键，
play_get_dtmf(&handle,"menu ","*0#",5,20,0,dtmf, member_id);
//不播音，直接等按键，
check_play_dtmf_event (&handle,"#",0,0,0,dtmf, member_id);
```

其他说明：

a) play_and_get_digits 无论设置啥条件，用户按键都会打断播音，假如你想要设置播音不被按键打断，请考虑使用下面的方法：

```
sprintf(cmd_tmp,"api    uuid_broadcast %s %s both",uuid,filename);
 esl_send_recv_timed(handle, cmd_tmp,1000);
```

这个 播音按键是无法打断的要停止这个 uuid_broadcast 的播音只能调用 uuid_break 来停止。

b) 另外测试还发现 启动 app:play_and_get_digits 会多出 1 秒时间,比如播 1 秒的语音从开始到结束需要 2 秒，因此改为 uuid_broadcast 可以提高播放 类似 198 的数字语音：1;b;9;s;8 的体验。

183. FreeSwitch ESL 开发如何支持录音取按键？

主要有 3 个函数组成，说明如下：

```
//事件体检查内容函数,这个上面已经介绍过
int check_play_record_body()
```

```
//录音取按键函数
```

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

```
//参数:handle:会话 handle
```

//uuid： 会话的 id

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//EndDtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//member_id 会议的 id >0 =0 表示没有进入会议

```
int record_get_dtmf(esl_handle_t *handle,char*uuid,char*filename,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int&member_id)
{
    int res;
    char cmd_tmp[1024],enddtmf[128],outdtmfbuff[128];
    if(EndDtmf==NULL || EndDtmf[0]==0)
        strcpy(enddtmf,"q");
    else
        strcpy(enddtmf,EndDtmf);
    if(outdtmf) outdtmf[0]=0;
    char *pan, _file[BUF_SIZE*2],tfile[BUF_SIZE*2];
    strcpy(_file,filename);
    pan=strstr(_file,":");
    if(pan)
    {
        strcpy(tfile, _file);
    }
    else
    {
        strcpy(tfile, recdir);
        strcat(tfile, "/");
        strcat(tfile, _file);
        if(strstr(_file,".")==0)
        {
            strcat(tfile,".alaw");
        }
    }
    CheckCreateDirectorys(tfile);
    if(_access(tfile,0)!=-1) remove(tfile);
    disp_msg("record_get_dtmf:%s",tfile);
    memset(outdtmfbuff,0,sizeof(outdtmfbuff));
    sprintf(cmd_tmp,"api uuid_record %s start %s %d",uuid,tfile,MaxTimer*2);
```

//MaxTimer*2 的条件是为了保证 uuid_record 的条件无效。而是通过自己的 ivr //check_record_dtmf_event 函数里面进行条件录音结束判断

```
        esl_send_recv_timed(handle, cmd_tmp,1000);
        res=check_record_dtmf_event(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtmfTimer,outdtmfbuff,member_id);
        if(outdtmf) strcpy(outdtmf,outdtmfbuff);
        disp_msg("record_get_dtmf:%s,end,dtmf=[%s]",tfile,outdtmfbuff);
        sprintf(cmd_tmp,"api uuid_record %s stop all",uuid);
        disp_msg("record_get_dtmf:%s stop",tfile);
        esl_send_recv_timed(handle, cmd_tmp,1000);
        disp_msg("record_get_dtmf:%s stop end",tfile);
        return res;
}
```

其他说明：默认 uuid_record 在 A 和 B 通话的时候是录制两个人的声音，

假如你要在通话的时候 A 这边只要录制 A 边的声音，请设置以下参数：

<action application="set" data="RECORD_READ_ONLY=true"/>

假如要在 A 这边录制 B 那边的声音，请设置以下参数：

<action application="set" data="RECORD_WRITE_ONLY=true"/>

//录音之后的取按键检查函数

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

//参数:handle:会话 handle

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//enddtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//其它说明:假如只有播音,不收取按键 设置：MaxDtmf=0

//member_id 会议的 id >0 =0 表示没有进入会议

```
int check_record_dtmf_event(esl_handle_t *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int& member_id)
{
 int done = 0,c=0,twodtmfc=0;
 esl_status_t status;
 time_t exp = 0;
 char dtmf[128];
 int res=0,press_dtmf;
 int  channel_execute=0;
 press_dtmf=1;
 if(outdtmf) outdtmf[0]=0;
 dtmf[0]=0;
 while((status = esl_recv_timed(handle, 1000)) != ESL_FAIL)
 {
        if(press_dtmf)
```

```
            {
                twodtmfc++;
                c++;
            }
            if(MaxTimer>0 && c>=MaxTimer){disp_msg("Waiting alldtmf %d seconds timout.",c); res=100;break;}
            if(TwoDtmfTimer>0 && twodtmfc>0 && twodtmfc>=TwoDtmfTimer){disp_msg("Waiting twodtmf %d seconds timout.",c); res=101;break;}
            if (status == ESL_SUCCESS)
            {
                const char *type = esl_event_get_header(handle->last_event, "content-type");
                if (type)
                {
                    if(strcasecmp(type, "text/disconnect-notice")==0)
                    {
                        const char *dispo = esl_event_get_header(handle->last_event, "content-disposition");
                        disp_msg("Got a disconnection notice dispostion: [%s]", dispo ? dispo : "");
                        if (dispo && strcmp(dispo, "linger")==0)
                        {
                            res=-99;
                            break;
                        }
                    }
                    if(strcasecmp(type, "text/event-plain")==0)
                    {
                        const char *eventbody=esl_event_get_body(handle->last_event);
                        int oldlen=strlen(outdtmf);
    if((res=check_play_record_body(handle,uuid,eventbody,outdtmf,MaxDtmf,enddtmf,&channel_execute,member_id)))
                        {
                            break;
                        }
                        if(oldlen==1) press_dtmf=1;
                        if(strlen(outdtmf)>oldlen)   twodtmfc=0;
                    }
                }
            }
        }
        return res;
}
```

使用例子如下 ：

// 录音到 z:/目录下 .alaw 扩展名表示 录制为 8K8bit 的 alaw 格式语音，.wav 扩展名录制为 8K16bit 的线性 pcm 格式语音

```
            sprintf(recordfilename,"z:/%s.alaw",uuid);//录制到 z:/目录下 ，以 uuid 作为文件名次
            record_get_dtmf(&handle,uuid,recordfilename,"#",1,30,0,dtmf,member_id);
```

184. FreeSwitch ESL 开发如何停止正在进行的播音录音等媒体操作？

停止录音要看是否是会议录音，假如是会议录音 停止方法如下：

```
if(member_id>0 && meetno>0 && recordmeet>0)
{
  sprintf(cmd_tmp,"%d recording stop all",meetno);
  disp_msg("conference %s",cmd_tmp);
  sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
  disp_msg(tmp);
  esl_send_recv(handle, tmp);
  if (handle->last_sr_event)
  {
        const char *err = NULL;
        if (handle->last_sr_event->body) {
              disp_msg("%s\n", handle->last_sr_event->body);
        } else if ((err = esl_event_get_header(handle->last_sr_event, "reply-text")) && !strncasecmp(err, "-err", 3)) {
              disp_msg("Error: %s!\n", err + 4);
        }
  }
  recordmeet=0;
  break;
}
```

否则假如是通道的播音和录音等媒体操作，停止的方法如下：

```
sprintf(cmd_tmp,"api uuid_break %s all",uuid);
esl_send_recv_timed(handle, cmd_tmp,1000);
```

185. FreeSwitch ESL 开发如何支持 ivr 电话呼叫通之后的连接和断开？

连接两个通道，在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"api uuid_bridge %s %s",uuid,uuid_to);
esl_send_recv_timed(handle, cmd_tmp,1000);
```

某个通道要断开和其它通道的媒体连接，在 mycallback 函数里面使用以下办法：

```
esl_execute(handle, "park","" , NULL);
```

其他说明：

ESL 的 uuid_bridge 执行之后，默认情况下用户挂机，坐席也会被挂机（下线）

需要做如下修改：

1.switch_ivr_bridge.c

static switch_status_t audio_bridge_on_exchange_media(switch_core_session_t *session)

函数里面的：

//yhy2013-10-22 uuid_bridge 之后会挂机。改为判断一下 是否需要挂机。配合 <action application="export" data="hangup_after_bridge=true"/>使用

```
if (switch_channel_test_flag(channel, CF_ANSWERED) && switch_true(switch_channel_get_variable(channel, SWITCH_HANGUP_AFTER_BRIDGE_VARIABLE)))
{
    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(session), SWITCH_LOG_ERROR, "yhy SWITCH_CAUSE_NORMAL_CLEARING\n");
    switch_channel_hangup(channel, SWITCH_CAUSE_NORMAL_CLEARING);
}
else
    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(session), SWITCH_LOG_ERROR, "yhy SWITCH_HANGUP_AFTER_BRIDGE_VARIABLE\n");
```

和

```
if (switch_channel_get_state(channel) == CS_EXCHANGE_MEDIA) {
    switch_channel_set_variable(channel, "park_timeout", "72000");//yhy2013-10-22 uuid_bridge 之后会挂机。改为 park 20hours
    switch_channel_set_state(channel, CS_PARK);
}
```

然后设置拨号计划里面的：

2.<action application="export" data="hangup_after_bridge=false"/>

3.然后 uuid_bridge 的时候 uuid_bridge 坐席 用户。

这样用户挂机，坐席就不会被挂机

186. FreeSwitch ESL 如何接收传真？

```
case OP_RECVFAX:
    if(strlen(userevent.filename)<=0)
    {
        disp_msg("index=%03d,OP_RECVFAX,no filename fail!",index);
        break;
    }
        esl_execute(handle, "playback", "silence_stream://2000", NULL);
        esl_execute(handle, "rxfax", userevent.filename, NULL);
    break;
```

说明：1.接收的文件必须是*.tif 格式的文件名。

2.拨号计划里面，socket 应用之前需要加上下面的配置以支持 T38 协议：

```
<action application="set" data="fax_enable_t38=true"/>
<action application="set" data="fax_enable_t38_request=true"/>
<action application="socket" data="127.0.0.1:8084 async full"/>
```

187. FreeSwitch ESL 如何发送传真？

```
case OP_ OP_SENDFAX:
        if(strlen(userevent.filename)<=0)
        {
            disp_msg("index=%03d,OP_RECVFAX,no filename fail!",index);
            break;
        }
        esl_execute(handle, "txfax", userevent.filename, NULL);
        break;
```

说明：发送的文件必须是*.tif 格式的文件。假如不是，需要先使用格式转换工具转换好才能发送。

188. FreeSwitch ESL 开发发送传真后如何获得传真的结果是成功还是失败？

在 hangup 是时候在 CHANNEL_HANGUP 中存在 FAX 中描述的变量。

result-code = 0 表示 OK。

189. FreeSwitch ESL 开发如何支持电话会议？

某个通道要参加会议，在 mycallback 函数里面使用以下办法：

```
sprintf(cmd_tmp,"%d@default",meetno);
esl_execute(handle, "conference",cmd_tmp , NULL);
```

其中的会议号码理论上没有任何限制。

会议的配置参数参照 conf\autoload_configs\conference.conf.xml 配置文件

默认的会议配置文件用户可以按键控制会场很多操作。

一般使用中我们全部屏蔽，修改如下：

```
<caller-controls>
    <group name="default">
    <!--        <control action="mute" digits="0"/>
        <control action="deaf mute" digits="*"/>
        <control action="energy up" digits="9"/>
        <control action="energy equ" digits="8"/>
        <control action="energy dn" digits="7"/>
```

```
    <control action="vol talk up" digits="3"/>
    <control action="vol talk zero" digits="2"/>
    <control action="vol talk dn" digits="1"/>
    <control action="vol listen up" digits="6"/>
    <control action="vol listen zero" digits="5"/>
    <control action="vol listen dn" digits="4"/>
    <control action="hangup" digits="#"/>   -->
    <control action="energy equ" digits="a"/>
  </group>
</caller-controls>
```

190. FreeSwitch ESL 开发如何支持退出电话会议？

退出电话会议在 mycallback 函数使用以下办法实现：

```
esl_execute(handle, "park","" , NULL);
```

191. FreeSwitch ESL 开发如何实现系统对会场的某个人静音和去除静音？

设置某个人静音（只能听，不能说）在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"%d mute %d",userevent.data,member_id);
sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
disp_msg(tmp);
esl_send_recv(handle, tmp);
if (handle->last_sr_event)
{
 const char *err = NULL;
 if (handle->last_sr_event->body) {
        disp_msg("%s\n", handle->last_sr_event->body);
 } else if ((err = esl_event_get_header(handle->last_sr_event, "reply-text")) &&
!strncasecmp(err, "-err", 3)) {
        disp_msg("Error: %s!\n", err + 4);
 }
}
```

设置某个人去除静音（能说话）在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"%d unmute %d",userevent.data,member_id);
sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
disp_msg(tmp);
esl_send_recv(handle, tmp);
if (handle->last_sr_event)
{
 const char *err = NULL;
```

```
if (handle->last_sr_event->body) {
        disp_msg("%s\n", handle->last_sr_event->body);
} else if ((err = esl_event_get_header(handle->last_sr_event, "reply-text")) &&
 !strncasecmp(err, "-err", 3)) {
        disp_msg("Error: %s!\n", err + 4);
}
}
```

说明：注意上面 的 api 命令里面的 console_execute: true
经过本人测试 没有这个，会议的设置不会起作用，这个的 console_execute: true 是参考 FS_Cli.c 的源码实现的。

192. FreeSwitch ESL 开发如何支持电话会议开始录音和停止录音？

电话会议录音，在 mycallback 函数使用以下办法实现：
开始会议录音：

```
disp_msg("OP_RECCONF:%s",tfile);
sprintf(cmd_tmp,"%d recording start %s",meetno,tfile);
recordmeet=1;
 sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
 disp_msg("index=%03d,api conference %s",index,cmd_tmp);
 esl_send_recv(handle, tmp);
```

停止会议录音：

```
sprintf(cmd_tmp,"%d recording stop all",meetno);
 recordmeet=0;
 sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
 disp_msg("index=%03d,api conference %s",index,cmd_tmp);
 esl_send_recv(handle, tmp);
```

其它说明：电话会议结束，会议录音自动结束。

193. 如何使用 FreeSwitch ESL 发送 IM 消息？

系统给分机发 IM 可以通过 ESL 来发：

```
 int send_stateReport_message(const char*from,const char*to,const char*chatmessage)
{
 int res=0,result=0;
 char tmp[4096];
 ESLevent *e;
 sprintf(tmp,"%d",fsport);
 if(con==NULL || con->connected()==false ) con= new ESLconnection(fsip, tmp, password);
 e = new ESLevent("custom", "SMS::SEND_MESSAGE");
```

```
        sprintf(tmp,"%s@%s",from,fschatserverip);
        e->addHeader("from", tmp);
        sprintf(tmp,"%s@%s",to,fschatserverip);
        e->addHeader("to", tmp);
        e->addHeader("type", "text/plain");//没有这个有些 sip 不行，收到了不显示
        e->addHeader("sip_profile", "internal");
        e->addHeader("dest_proto", "sip");
        e->addBody(chatmessage);
        con->sendEvent(e);
        disp_msg("from[%s]==>to[%s],data=[%s]",from,to,chatmessage);
        delete e;
        return 0;
}
```

其它说明：

假如要发送汉字字符串，需要先转换成 UTF-8 格式之后才能发送。直接发送 GBK 或者 GB2312 的内码，对方收到的会是乱码。

比如 汉字“呼叫过你” 需要先转换为转换为 UTF-8 格式 ：“鍛煎彫杩囦綘”然后发送。对方收到的才会显示汉字。假如对方收到的不是汉字，是 UTF-8 的乱码“鍛煎彫杩囦綘”说明对方不支持汉字。

194. 如何使用 FreeSwitch ESL 开发的完整 IVR 演示流程？

系统提供的 demo 比较简单，没有处理基本的 ivr 的工作单元：

播音取按键，录音取按键，电话转接，参加会议

下面的这个文件能实现上面的这些基本功能：

先修改拨号计划：增加下面的内容：

```
        <extension name="socket">
        <condition field="destination_number" expression="^12396$">
        <action application="set" data="call_timeout=30"/>
        <action application="set" data="record_sample_rate=8000"/>
        <action application="export" data="RECORD_STEREO=false"/>
        <action application="set" data="hangup_after_bridge=false"/>
        <action application="set" data="continue_on_fail=true"/>
        <action application="socket" data="127.0.0.1:8084 async full"/>
        </condition>
        </extension>
```

然后使用软电话注册上去然后拨打 12396,完整的 main.cpp 代码请参照附录下载

195. FreeSwitch 的 ESL 开发如何才支持 CSP（连续语音流）模式的 ASR 语音识别和 SVR 声纹识别？

使用 FreeSwtich 做 ASR 和 SVR 有几种方法：

1.是使用离线方式，首先录制语音到文件里面然后对文件进行离线语音识别识别。

这个方式现在已经 OUT 了，不对应该说很早就已经 OUT 了。缺点就不说了，大家想象一下都知道。DISK IO 大，用户体验差。

2.使用 mod_unimrcp 模块的客户端基于 MRCP2.0 的客户端做的。这个一来要求服务端引擎要支持 MRCP2.0，二来只能支持 ASR，不支持 SVR。

3.使用 mod_pocketsphinx 模块进行 ASR，这个是一个美国的卡内基梅隆大学搞的开源的 ASR，

具体使用参考：http://wiki.freeswitch.org/wiki/Mod_pocketsphinx

根据我的测试，使用 js 的脚本，识别英文啥的没有问题，不过电话信道的识别率有待考证。另外中文识别也是一道坎，SVR 声纹识别不支持。

4.自己折腾的其他方法。

由于 1，2，3 都不是完美的解决方案，因此决定采用 4.自己折腾的其他方法。

开始自己折腾，由于本人之前做过 Nuance 和中科信利的 ASR 语音识别和天聪的声纹识别引擎的应用开发工作。开发集成相对来说工作量和难道不大。

下面详细介绍这个方案

首先是系统构架。具体参见“基于 CTI 的平台软件体系结构”

其次是连续的语音流的获取：

语音识别的核心是语音流，FreeSwtich 的语音流从 RTP 出来之后通过共享内存写到内存里面，然后识别引擎的客户端从内存里面获取到用户的语音流，发给识别引擎进行识别。经过评估，本人决定在 mod_pocketsphinx 模块上修改

然后是代码的修改。

把 mod_pocketsphinx 改为通用的 ASR/SVR 模块：修改 mod_pocketsphinx.c 源码， 将收到的 RTP 语音流写入到对应的通道的共享内存里面，其他的识别引擎的客户端从共享内存里面获取对应的通道的语音 。

主要修改到的地方如下：

```
SWITCH_MODULE_LOAD_FUNCTION(mod_pocketsphinx_load)
{
    switch_asr_interface_t *asr_interface;
    switch_mutex_init(&MUTEX, SWITCH_MUTEX_NESTED, pool);
    globals.pool = pool;
    if ((switch_event_bind_removable(modname, SWITCH_EVENT_RELOADXML, NULL, event_handler, NULL, &NODE) != SWITCH_STATUS_SUCCESS)) {
        switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "Couldn't bind!\n");
    }
    do_load();
    InitShareRam(1); //初始化共享内存
    。。。
```

```
SWITCH_MODULE_SHUTDOWN_FUNCTION(mod_pocketsphinx_shutdown)
{
	switch_event_unbind(&NODE);
	CloseShareRam(1);//关闭共享内存
	return SWITCH_STATUS_UNLOAD;
}


/*! function to load a grammar to the asr interface */
static switch_status_t pocketsphinx_asr_load_grammar(switch_asr_handle_t *ah, const char *grammar, const char *name)
{
	char *jsgf, *dic, *model, *rate = NULL;
	pocketsphinx_t *ps = (pocketsphinx_t *) ah->private_info;
	switch_status_t status = SWITCH_STATUS_FALSE;

	switch_log_printf(SWITCH_CHANNEL_LOG,
SWITCH_LOG_WARNING,"asr_start,load_grammar=%s,name=%s\n",grammar,name);
	switch_safe_free(ps->grammar);
	ps->score=0;
	ps->confidence=0;
	ps->grammar = strdup(grammar);//语法内容实际上是需要写入的内存的通道编号
	return SWITCH_STATUS_SUCCESS;
}

/*! function to feed audio to the ASR 收到语音流 */
static switch_status_t pocketsphinx_asr_feed(switch_asr_handle_t *ah, void *data, unsigned int len, switch_asr_flag_t *flags)
{
	char s[2048];
	pocketsphinx_t *ps = (pocketsphinx_t *) ah->private_info;
	int rv = 0,i,index=0;
	short*p=data;

	if(ps->grammar)
		index=atoi(ps->grammar);
	else
		index=-1;
	 rv=WriteRam(index,ps->score,(char*)data,len);//写入到共享内存
	if(rv>0)
	{
		ps->score=rv;
	}
	else
	{
```

```
        //      switch_log_printf(SWITCH_CHANNEL_LOG,                SWITCH_LOG_WARNING,"index=%d,pos=%d,len=%d,WriteRam
fail!\n",index,ps->score,len);
        }
        return SWITCH_STATUS_SUCCESS;
}

int InitShareRam(int argc)
{
        char filename[MAX_RAM_FILE_NUMBER][32],
                ramname[MAX_RAM_FILE_NUMBER][32],
                tmp[200];
        int i,openflag=2,res;

        if(argc!=1)
                openflag=RAM_READ;
        else
                openflag=RAM_WRITE;
        ...
}
int    CloseShareRam(int argc)
{
        int i,openflag;
        char filename[MAX_RAM_FILE_NUMBER][32];

        if(argc!=1)
                openflag=RAM_READ;
        else
                openflag=RAM_WRITE;
        for(i=linedev_start1;i<=linedev_end1;i++)
        {
                sprintf(filename[i],"%s/temp/asr01%04d.pcm",sharerecdir,i);
                close_ramfile(openflag,i,filename[i]);
        }
        return 0;
}
int WriteRam(int fileno,int pos,char*buffer,int size)
{
        static long count=0L;
        char tmp[100];
        int s=0,all=0,p=0;
        int headsize=4;

        if(fileno<linedev_start1 || fileno>linedev_end1) return -1;
        if(pRam[fileno]<=0) return -1;
```

```
	if(pos<0 || (pos+headsize)>=MAX_RAM_SIZE || size<=0) return -1;
	s=size;
	if(s+headsize>=MAX_RAM_SIZE)
		s=MAX_RAM_SIZE-headsize;
	else
		if(pos+s+headsize>=MAX_RAM_SIZE)
			s=MAX_RAM_SIZE-headsize-pos;
	if(s<=0) return -1;
	p=pos+headsize;
	all=p+s;
	*(int *)tmp=all;
	memcpy((char*)(pRam[fileno]),tmp,headsize);
	memcpy((char*)(pRam[fileno])+p,buffer,s);
	return pos+s;
}
```

最后是 ESL 代码的开发。

```
 case OP_PLAYASRDTMF://播音同时进行语音识别和取按键
{
	dtmf[0]=0;
	flush_status(index, "AsrPlay" );
	sprintf(cmd_tmp,"pocketsphinx %d    undefined",index);
	esl_execute(handle, "detect_speech", cmd_tmp, NULL);
	disp_msg("index=%03d,play_asr_dtmf start,file=%s",index,userevent.filename);
	res=play_get_dtmf(index,handle,uuid,userevent.filename,userevent.EndDtmf  ,userevent.MaxDtmf,userevent.MaxTimer,use
revent.TwoDtmfTimer,dtmf,member_id);
	disp_msg("index=%03d,play_asr_dtmf end",index);
	if(res<0) {esl_execute(handle, "detect_speech", "stop", NULL);user_disconnect=1;break;}
	if(res==OP_DROPCALL) {esl_execute(handle, "detect_speech", "stop", NULL);dropcall=1;break;}
	if(res==OP_STOPPLAY)
	{
		flush_status(index, "StopPlay" );
		sprintf(cmd_tmp,"api uuid_break %s all",uuid);esl_send_recv_timed(handle, cmd_tmp,1000);
		send_media_event(index,EV_PLAYOVER,dtmf);
	}
	else
		send_media_event(index,EV_VOCDIALNUM,dtmf);
	flush_status(index, "AsrRec" );
	disp_msg("index=%03d,get_dtmf
start,EndDtmf=%s,MaxDtmf=%d,MaxTimer=%d,TwoDtmfTimer=%d",index,userevent.EndDtmf ,userevent.MaxDtmf,userevent.Ma
xTimer,userevent.TwoDtmfTimer);
	res=check_play_dtmf_event(index,handle,uuid,userevent.EndDtmf,userevent.MaxDtmf,userevent.MaxTimer,userevent.Two
DtmfTimer,dtmf,member_id,2,userevent.MaxTimer);
```

```
        disp_msg("index=%03d,get_dtmf end",index);
        esl_execute(handle, "detect_speech", "stop", NULL);//stop record
        if(res<0) {user_disconnect=1;break;}
        if(res==OP_DROPCALL) {dropcall=1;break;}
        if(res==OP_ECSTOP||res==OP_STOPPLAY||res==OP_STOPRECORD){flush_status(index, "StopRec" );  sprintf(cmd_tmp,"api
uuid_break %s all",uuid);esl_send_recv_timed(handle, cmd_tmp,1000);}
        send_media_event(index,EV_PLAYASROVER,dtmf);
        break;
}

case OP_RECORDCSP://通话过程对 1 个通道进行 csp 录音，进行声纹识别
{
        sprintf(cmd_tmp,"pocketsphinx %d    undefined",index);
        esl_execute(handle, "detect_speech", cmd_tmp, NULL);
        //esl_execute(handle, "detect_speech", "resume", NULL);
        flush_status(index, "RecCSP" );
        dtmf[0]=0;
        if(userevent.EndDtmf[0]==0 && userevent.MaxDtmf==0 && userevent.MaxTimer==0 && userevent.TwoDtmfTimer==0)
userevent.MaxDtmf=30;
        disp_msg("index=%03d,get_recordcsp_dtmf
start,EndDtmf=%s,MaxDtmf=%d,MaxTimer=%d,TwoDtmfTimer=%d",index,userevent.EndDtmf ,userevent.MaxDtmf,userevent.Ma
xTimer,userevent.TwoDtmfTimer);
        res=check_play_dtmf_event(index,handle,uuid,userevent.EndDtmf,userevent.MaxDtmf,userevent.MaxTimer,userevent.Two
DtmfTimer,dtmf,member_id,2,userevent.MaxTimer);
        disp_msg("index=%03d,get_recordcsp_dtmf end",index);
        if(res<0) {user_disconnect=1;break;}
        if(res==OP_DROPCALL) {dropcall=1;break;}
        send_media_event(index,EV_PLAYASROVER,dtmf);
        esl_execute(handle, "detect_speech", "stop", NULL);
        send_media_event(index,EV_RECOVER,dtmf);
        break;
}
```

其他说明：

上面使用共享内存的技术，这种共享内存的技术在进程间通信的大数据量的速度上是最快的。

但是由于虚拟机上的进程间通信无法使用共享内存的技术，因此假如 FreeSwitch 运行在虚拟机上，将导致无法使用此功能。假如不幸出现这种情况，可以考虑将共享内存改为使用 UDP 或者消息进行进程间通信。

196. FreeSwitch 如何实现彩铃功能？

拨号计划设置彩铃的方法：

```
<extension name="Local_Extension2">
  <condition field="destination_number" expression="^([0-9]\d+)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="set" data="call_timeout=30"/>
        <action application="export" data="hangup_after_bridge=true"/>
        <action application="set" data="continue_on_fail=true"/>
         <action application="export" data="record_sample_rate=8000"/>
         <action application="export" data="RECORD_STEREO=false"/>
      <action application="set" data="ringback=d:/data/system/music_erge.alaw"/>
        <action application="bridge" data="user/$1@${domain_name}"/>
         <action application="hangup"/>
      </condition>
</extension>
```

ESL 要实现彩铃需要下面 3 个步骤
用户的来电先回复 183，这个需要
在拨号计划或者 esl 里面执行 pre_answer，
然后 esl 给主叫播放彩铃音乐，
然后呼叫被叫，被叫接通，停止播放音乐，把主叫个被叫使用 uuid_bridge 连接起来。

197. FreeSwitch ESL 如何实设置和获取通道变量功能？

esl 里面 set 和 get 通道变量：
设置：

```
esl_execute(handle, "set", "hangup_after_bridge=false", NULL);
```

获取：

```
int check_var(esl_handle_t *handle,char*uuid,char *var_name)
{
    int res=0,result=0;
    char tmp[128];
    disp_msg("check_var");

    sprintf(tmp,"api uuid_getvar %s   %s\n\n",uuid,var_name);
    res=esl_send_recv_timed(handle, tmp,1000);
    if(res!=ESL_SUCCESS) //fs broken
    {
        return -2;
    }
    if (handle->last_sr_event && handle->last_sr_event->body)
    {
        disp_msg("check_var:[%s][%s]\n",var_name,handle->last_sr_event->body);
```

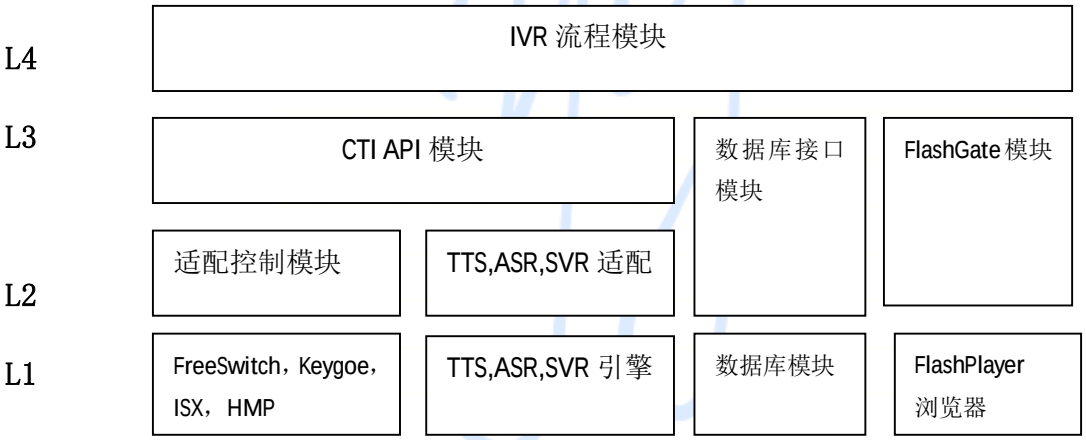
```
        }
        disp_msg("check_var end");
        return result;
    }
```

第十一章 CTI 平台开发部分

这个部分目标是基于 FreeSwitch 和 FreeSwitch 的 ESL 开发一个带 API 接口的 CTI 平台。本人称之为 CTI 平台,开发一个平台的目的是为了适配多种软交换的硬件，而不仅仅是支持 FreeSwitch。理由是因为不同的软交换功能和性能不一样，比如 FreeSwitch 会议混音功能比较差，比如 FreeSwitch 视频会议暂时不支持，等等，这个使用就需要其它软交换或者多媒体交换机的支撑。在一个平台内能使用统一的 API 来开发，而不管底层的各种具体型号的软交换或者多媒体交换机的差异，这个就是平台的作用。

198. CTI 的平台层次结构？

CTI 平台的层次结构如下：



系统设计为 4 层层次结构.

L1 为底层资源层，支持多种硬件平台，支持多种引擎，各种数据库

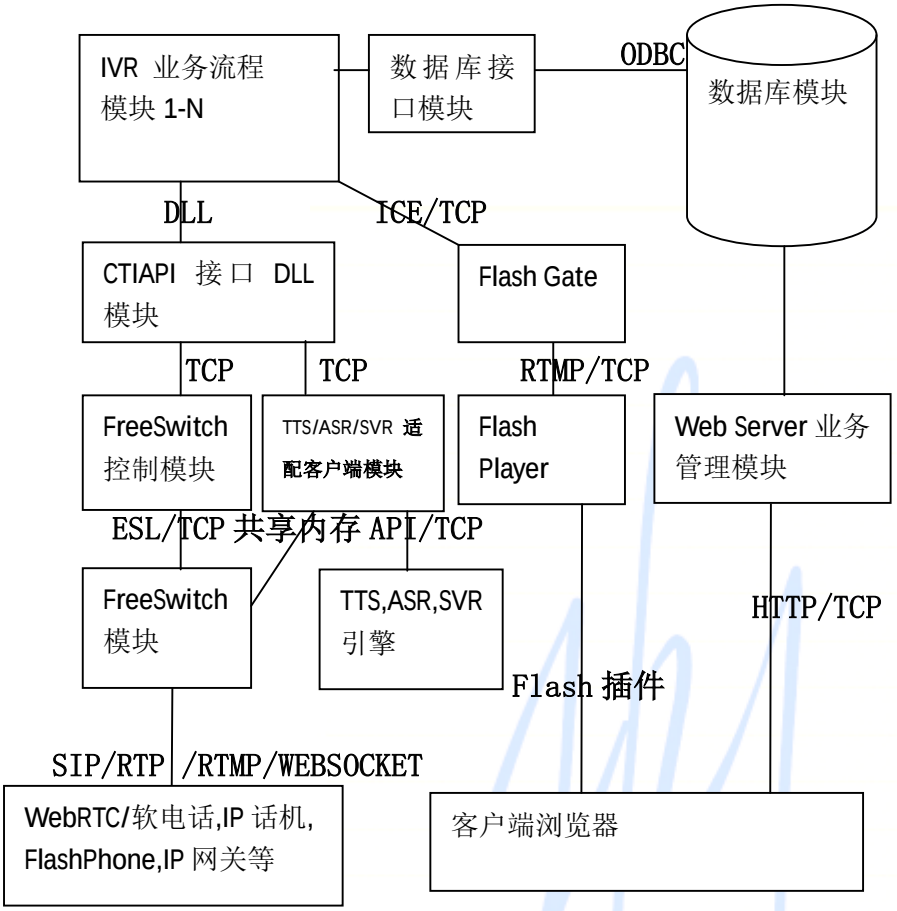
L2 为资源控制层，包括各种硬件的适配和引擎的适配，数据库接口 ODBC 模块

L3 为接口层，CTIAPI.DLL

L4 为应用层 ，IVR 业务流程，

199. 基于 CTI 的平台软件体系结构？

基于 FreeSwitch 的 CTI 开发的平台的软件体系结构如下：



说明:1.IVR 流程模块通过 CTIAPI.DLL 跟 FreeSwitch 控制模块通信,
进而通过 ESL 控制 FreeSwitch 的 sip 和媒体操作

2. IVR 流程模块通过 ICE 中间件跟 java 编写的 Flash Gate 模块数据通信.
3. IVR 流程通过通过 ODBC 访问数据库
4. 客户端模块通过浏览器里面的 flash 插件与 Flash Gate 模块数据通信.
5. 客户端模块通过 SIP/RTP 协议与 FreeSwitch 模块进行媒体通信.
6. 客户端模块通过 HTTP 访问系统的 Web Server 业务管理模块。
7. web 业务管理模块与数据库进行通信.
8. CTIAPI.DLL 通过 TCP 跟 TTS，ASR，SVR 模块进行数据通信。
9.TTS，ASR，SVR 模块控制具体的 TTS，ASR，SVR 引擎。

其它说明：
系统支持多个 IVR 业务流程模块，进而支持不同的业务功能，比如呼叫中心，电话会议，语音短信，网络传真等业务。IVR 业务流程模块是通过 CTIAPI.DLL 提供的接口对系统进行操作的。
TTS：文本到语音的转换

ASR：自动语音识别，也叫语音识别
SVR：说话人确认和识别，也叫声纹识别。
关于 ice 中间件:
Ice 是 Internet Communications Engine 的简称，ICE 是 ZeroC 提 供的一款高性能的中间件，基于 ICE 可以实现电信级的解决方案。
ICE 是一种面向对象的中间件平台，支持面向对象的 RPC 编程，其最初的目的是为了提供类似 CORBA 技术的强大功能，又能消除 CORBA 技术的复杂性。该平台为构建面向对象的客户－服务器应用提供了工具、API 和库支持。
由 ICE 平台开发的应用支持跨平台部署，多语言编程，其中服务端支持 C++、JAVA、C#、Python 等几种编程语言，客户端还支持 Ruby、PHP。ICE 支持同步/异步、订阅/发布的编程模式，支持分布式部署，网格计算，内置负载均衡功能，支持 SSL 安全加密。

关于 ice 中间件的详细介绍参见：
http://www.zeroc.com/ice.html

200. CTI API 的平台软件模块总共有几个？

整个平台包括 fs esl 控制适配模块 fsg，ag 会话管理模块，tts 模块，api dll 5 个底层平台模块
动态流程 dyflow，坐席控制 acd，群呼模块 notify，flashgate，flash 坐席控制插件，坐席话机外拨 dialout 模块 6 个上层业务模块。业务模块根据配置选择支持 mysql 或者 oracle 数据库。
整个平台是基于 windows 开发的，底层模块可以支持 linux 版本。
底层平台模块的接口是 C API。
用户可以使用 API 这个自己开发其他业务，比如电话会议功能，电子传真功能等等。
上层业务模块的接口是数据库和 JS API。用户可以使用 JS API 方便地集成到自己 web 系统中。

下面是一个投产系统的运营模块部署

下面这个是该系统 600 线外拨的电话营销系统的 fsg 的界面：

假如你刚好需要类似的呼叫中心和电话营销平台可以联系本人。

201. CTI 平台开发接口 API 如何设计？

要做到开发简单业务功能，FreeSwitch ESL 的平台的
CTIAPI.DLL 提供的接口必须满足几点：

1. 业务核心流程开发是单线程操作。
2. 业务流程开发里面的媒体操作必须是同步操作。

媒体操作必须是同步操作举例说明： 比如播音操作，假设语音是 60 秒再加上等用户按键 10 秒，同步操作意思就是播音一开始函数就阻塞了，一直到播音结束或者播音过程中用户的按键达到了条件。函数才返回，代码继续执行。
这样的流程开发比较容易理解，普通大学毕业生培训一星期就可以开发 ivr 业务流程。

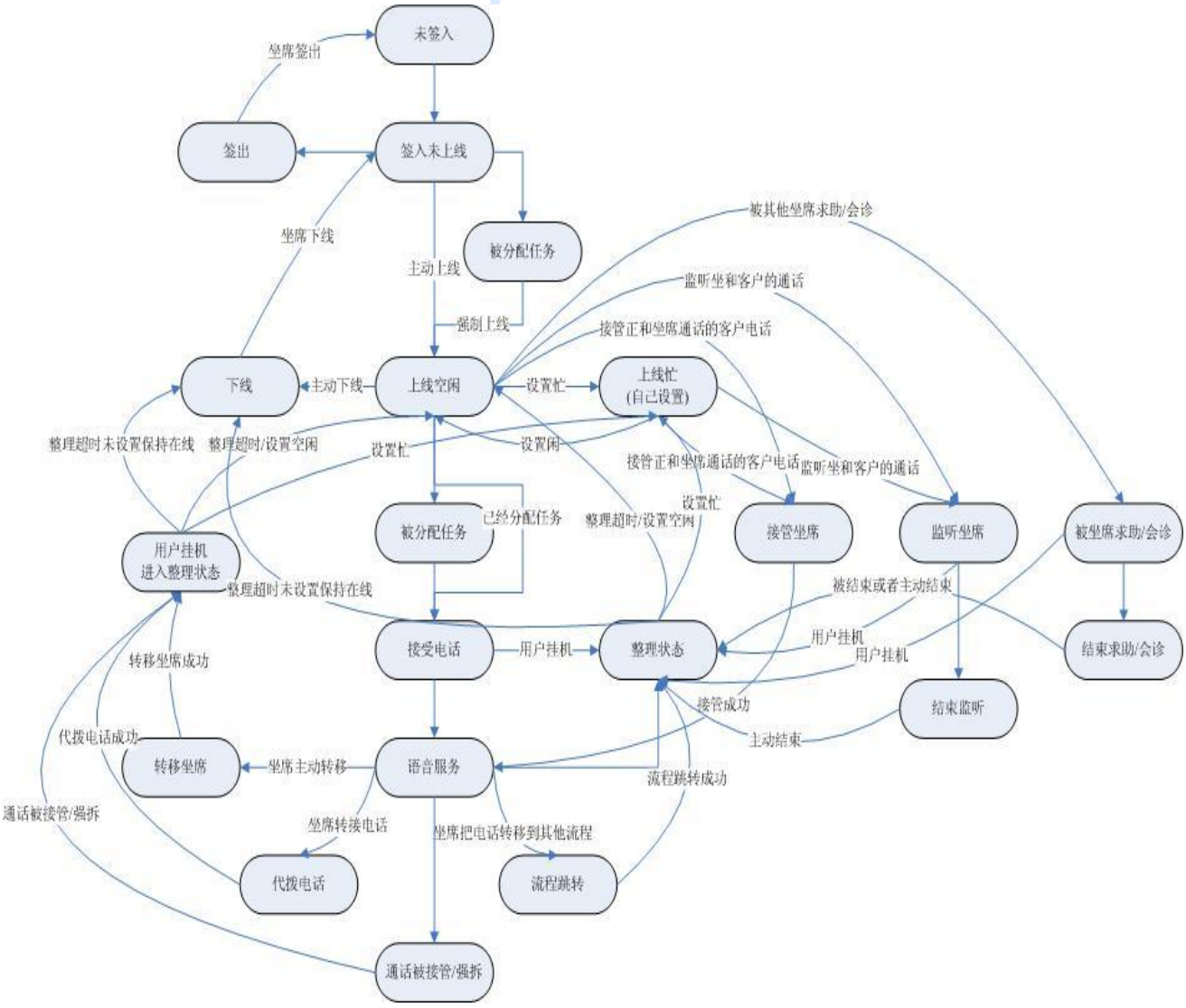
为了达到上面的要求，在 Windows 引入纤程开发机制。
Linux 下也有类似的机制。
具体开发设计的思想描述如下：

现实计算机系统中,大多系统可以以有限状态机进行描述.
有限状态机（英语：finite-state machine, FSM），又称有限状态自动机，简称状态机，是表示有限个状态以 及在这些状态之间的转移和动作等行为的数学模型.
FSM（有限状态机）可以使用使用多种类型的状态转移表。下面展示最常见的表示：当前状态（B）和条件（Y）的组合指示出下一个状态（C）。完整的动作信息可 以只使用脚注来增加。包括完整动作信息的 FSM 定义可以使用状态表。

状态转移表

当前状态 → 条件 ↓	状态 A	状态 B	状态 C
条件 X	...	...	...
条件 Y	...	状态 C	...
条件 Z	...	...	...

有限状态机 FSM 的实现.通常是以 异步事件促发机制.来实现的.
具体来说处理过程如下,语音系统播放提示语音,用户在电话上按键,语音交换机或者语音板块检测到按键之后,停止播音,同时在按键输入条件达到之后将收到的按键通过事件的方式上报给语音流程控制程序.语音流程控制程序用户根据按键进行选择.在不同的状态下.根据不同的按键转到不同的状态上.
下面来看一个呼叫中心中坐席的 FSM 状态图
（你要是看的不晕说明有开发 FSM 的天赋 J）：



这个有限状态机就比较复杂.实际编程实现过程,使用异步模式编程,加上用户的拨入拨出控制,整个呼叫中心用户拨入,呼叫用户和坐席控制代码量达到了 528KB.
问题的提出:这种系统通常是一个语音流程要对外提供成百上千人(线)的语音服务.在某个时刻,这些成百上千人(线)处于不同的状态.要维护好这些成百上千人(线)不同的状态.再根据不同的事件转到不同的其它的状态.复杂度是很高的.

难点在于:
1) 并发用户多
2)用户状态多
3) 异步事件处理复杂
解决办法:
Widnows 下提供 fiber 纤程的一组实现 API:
CreateFiber()
ConvertThreadToFiber()
SwitchToFiber()
DeleteFiber()
GetFiberData();
GetCurrentFiber();

DLL 里面的核心函数介绍如下：
主要工作线程函数如下：

```
unsigned __stdcall ThreadVoiceFlow(void *x )
{
        ThreadFunc = ConvertThreadToFiber(0) ;
        if(ThreadFunc==NULL) cti_log(2,"ConvertThreadToFiber fail,stack size error.") ;

 for(int i=0;i<MaxCalls;i++)
 {
        FiberProcCallFunc[i] = CreateFiber(DEF_STACK_SIZE , FiberProcCall ,(void *) (i+1)) ;
        if(FiberProcCallFunc[i]==NULL)     cti_log(2,"Create FiberProcCallFunc[%d] fail,stack size error.",i) ;
 }

      FiberMain = CreateFiber(DEF_STACK_SIZE , FiberProcMain , NULL) ;
 if(FiberMain==NULL)   cti_log(2,"Create FiberMain fail,stack size error.") ;

      SwitchToFiber(FiberMain) ;//调度到主工作 仟程
      DeleteFiber(FiberMain);//删除   必须自己手动清理
      cti_log(2,"ThreadVoiceFlow main fiber end") ;
      return 0;
};
```

IVR 事件处理函数如下，里面有回调用户初始化传入的回调函数：

```
VOID WINAPI IvrEvent(TUCEvent evt,int selfid)
{
 char sid[128];
```

```
int index,evttype=evt.event;
try
{
	strcpy(sid,evt.sid);
	index=atoi(sid);

	//call in
	if(evttype==EV_OFFERED)
	{
		return ;
	}
	if(evttype==EV_ACCEPT)//call in
	{
		if(index>=0 && index<MaxCalls)  {linestate[index]=1;lineconnectstate[index]=1;}
		if(ringconnect_callback_function) ringconnect_callback_function(sid,1);
		CTIDropCall(sid);
		if(index>=0 && index<MaxCalls)  {linestate[index]=0;lineconnectstate[index]=0;}
		cti_log(2,"IvrEvent call in session end,sid=%s",sid);
	}

	//call out
	if(evttype==EV_ALERTING)
	{
		if(index>=0 && index<MaxCalls)		{linestate[index]=1;lineconnectstate[index]=2;}
		if(ringconnect_callback_function) ringconnect_callback_function(sid,2);
		return ;
	}

	if(evttype==EV_CONNECTED)
	{
		if(index>=0 && index<MaxCalls)		{linestate[index]=1;lineconnectstate[index]=3;}
		if(ringconnect_callback_function) ringconnect_callback_function(sid,3);
		CTIDropCall(sid);
		if(index>=0 && index<MAX_INDEX)		{linestate[index]=0;lineconnectstate[index]=0;}
		cti_log(2,"IvrEvent call out session end,sid=%s",sid);
	}

	if(evttype==EV_DISCONNECTED)
	{
		if(index>=0 && index<MaxCalls)
		{
			if(ringconnect_callback_function) ringconnect_callback_function(sid,4);
			CTIDropCall(sid);
			linestate[index]=0;
```

```
                lineconnectstate[index]=0;
                cti_log(2,"IvrEvent call out session end4,sid=%s",sid);
            }
            return ;
        }

 }
 catch (...) {
 }
 return ;
}
```

工作的纤程函数如下：

```
VOID WINAPI FiberProcCall( PVOID lpParameter )
{
        int selfid=(long)lpParameter;
        cti_log(3,"FiberProcCall start, selfid=%d",selfid-1) ;
        while (1)
        {
            IvrEvent(_ivrevent,selfid);
            SwitchToFiber(FiberMain);//调度到主函数
          }
          cti_log(3,"FiberProcCall end") ;
        SwitchToFiber(FiberMain) ;
}

VOID WINAPI FiberProcMain( PVOID lpParameter )
{
        int result = 0,index,res,event,eventdata;

        cti_log(3,"FiberProcMain start") ;
      while (1)
        {
        index = pop_flow_event(&event, &eventdata);
        if(index<0)       //等待事件。0 表示有事件,其它表示没有
        {
            _ivrevent.event=0;
            if((result=pop_event(_userevent))<0)//>=0 表示有事件,其它表示没有
            {
                Sleep(1);
            }
            else    ////处理 user  自己设置的  event
            {
                res=ProcessUserMsg(_userevent);
                if(res) Sleep(1);
```

```
        }
        continue;
}

if(event==EV_NULLINDEX && eventdata==USERDIAL_STATE)
{
        cti_log(2,"get EV_NULLINDEX,sid=%d",index);
        if(wait_callout_index<0)
        {
                if(index==MAX_DEV)
                        wait_callout_index=-12;
                else
                        wait_callout_index=index;
        }
        SetEvent(m_hEvent);
        continue;
}

if((eventdata == OP_ADDCONF || eventdata == OP_SETCONF) &&
        (event == EV_OPSUCCESS || event==EV_OPFAILURE))
{
        if(addconf_result[index]==1)
        {
                addconf_result[index]=event;
                if(eventdata == OP_ADDCONF) SetEvent(m_hEventAddConf);
                if(eventdata == OP_SETCONF) SetEvent(m_hEventSetConf);
                continue;
        }
}
if(event==OP_CHANGE_TTS_OK || event==OP_CHANGE_TTS_FAIL)
{
        if(tts_result[index]==1)
        {
                tts_result[index]=event;
                SetEvent(m_hEventTTS);
                continue;
        }
}
sprintf(_ivrevent.sid,"%d",index);
_ivrevent.event=event;
_ivrevent.data=eventdata;
_ivrevent.time=time(NULL);
_ivrevent.msg[0]=0;
```

```
        if(_ivrevent.event==EV_OFFERED || _ivrevent.event==EV_ALERTING)//拨出和拨入的呼叫,创建
        {
            res=AllocFiberId(_ivrevent.sid);
            if(res<0)
            {
                cti_log(2,"AllocFiberId    CTIDropCall.sid=%s",_ivrevent.sid);
                CTIDropCall(_ivrevent.sid);
                continue;
            }
        }
        index=GetFiberIdBySessionId(_ivrevent.sid);

        if(_ivrevent.event==EV_DISCONNECTED && index>=0 && index<MaxCalls)      linestate[index]=0;

        if(index>=0 && index<MaxCalls)
        {
            cti_log(4,"SwitchTo sid=%d, event=%d,eventdata=%d,fun=0x%x",
            index,event,eventdata,FiberProcCallFunc[index]);
            SwitchToFiber(FiberProcCallFunc[index]);//调度到执行流程函数
        }
        if(Stop) break;
      }
 cti_log(2,"FiberProcMain end") ;
 SwitchToFiber(ThreadFunc) ;//回到线程
}
```

202. CTI 平台支持哪些硬件？

平台提供 API 的好处是 IVR 业务流程开发的人可以不管硬件的细节。
平台可以适配不同的硬件，
因为不同的硬件特长不一样，性价比也不一样。这样选择的余地就很大。
CTIAPI.DLL 不仅仅支持 FreeSwitch ，
也支持 Dialogic 的 R4，AG ，CG 等老卡，SPCI4 和 TX 系列老信令卡和 HMP 系列
也支持东进的 DN 系列老卡和 Keygoe 系列媒体交换机
也支持毅航 ISX 系列媒体交换机从 1000 到 6000 系列。其中 6000 系列最大单机 194E1（两个光纤（每个 63E1）+64 个 E1 口）.

203. CTI 平台 DLL 开发接口有哪些功能？

CTIAPI.DLL 提供基本的呼叫控制，媒体播音录音处理，会议控制，传真操作，声纹操作等功能。
CTIAPI.DLL 提供的具体接口以及说明如下：

```
/*
Write By yhy
*/
#ifndef __API_CTIApi_H
#define __API_CTIApi_H


#include <windows.h>
//--------------------------------------------------------------------------
#ifdef __cplusplus
extern "C" {
#endif

#ifdef CTIAPI_EXPORTS
#define DLLAPIEXP __declspec(dllexport)
#else
#define DLLAPIEXP __declspec(dllimport)
#endif
#define CTIAPI    WINAPI
/*错误代码说明:
-8:未调用 CTIOpenIvr()
-9:该函数只能在 callback_fun 里面调用,不能在其它 callback_fun 外部调用
-7:已经调用 CTIOpenIvr()，不能再次调用 CTIOpenIvr()，需要先调用 CTICloseIvr()。
-1:sessionId,会话 ID 为 NULL 或者="",错误。
*/
//用户提供的回调函数，当用户振铃，或者用户接通的时候 CTIAPI 会调用这个函数。用户在这个函数的 ring_connect=1 的时候写 ivr 流程。
//参数:sid  会话 id
//参数:ring_connect:1  用户拨入系统,系统振铃,未应答
//                    :2  系统拨出,收到用户振铃
//                    :3  系统拨出,收到用户接通
typedef int (CTIAPI * CTI_CALLBACK_FUN)(char* sid,int ring_connect);

//-------------------------------------//初始化,控制类 API-------------------------------
//初始化 Ivr
//返回:成功:返回  0.其它非零:错误.具体:-2:配置文件不存在,-3:callback_fun  为  NULL,-4:配置文件中的线路配置错误,-5:函数初始化多次.
//说明:该函数在程序启动的时候调用一次.调用之后,才能执行其它的函数.
DLLAPIEXP int CTIAPI CTIOpenIvr(char *configfile,CTI_CALLBACK_FUN callback_fun);
//退出 Ivr
//参数:无
//说明:该函数在程序退出的时候调用一次.
DLLAPIEXP int CTIAPI CTICloseIvr();
```

//说明:以下 API 函数的通用返回值具体说明:
///////返回:-99:收到挂机,-1:其它异常事件,-2:会话 id=NULL 或者="",-3:会话 id 越界<0 或者>最大线路,
////////////-4:会话 id 和系统保存的不匹配,-8:未初始化 Ivr.
//----------呼叫类 API----------
//应答
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
DLLAPIEXP int CTIAPI CTIAnswer(char* sessionId);
//挂机
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
DLLAPIEXP int CTIAPI CTIDropCall(char* sessionId);
//发起呼叫
//返回:成功:返回 0.其它非零:除通用返回值之外,其它错误.:-4:被叫为空，-5：主叫或者源被叫为 NULL，-10 外拨 500ms 超时没有收到回应.可以尝试重复调用,-11:指定线路呼叫的被叫号码指定的线路不在线路范围,-12:外拨没有空闲线路
//参数:sessionId:会话 id(输出参数)
//caller:主叫号码
//called:被叫号码
//callop:源被叫号码
//calltype:呼叫类型，拨入类型是 1，拨出>1,用户自己设置，将来可以通过 CTIGetCallType()获取到，用来区分不同的业务外拨,=99 的时候表示是指定线路外拨,被叫号码即为线路号,>=1000 的时候 CTIGetCallType()取回的 calltype=1000+sid
DLLAPIEXP int CTIAPI CTIMakeCall(char* sessionId,char*caller,char*called,char*callop,int calltype);
//----------交换操作类 API----------
//电路通话操作
//返回:成功:返回 0.其它非零:错误.
//参数:sid1:会话 id1 接收的(假如半双工，对应听的)会话
//sid2:会话 id2 发送的(假如半双工，对应说的)会话
//type:1:half 半双工,0:全双工 full,-1:unlisten,-2 free voice
//说明:半双工的时候是 sid2-->sid1,即 sid1 偷听 sid2 说,sid2 听不见 sid1 说.
//假如 sid1==sid2 表示是和语音连接(相当于申请语音资源).通常情况下,做媒体操作,系统会自动申请语音资源.不需要这个操作.
DLLAPIEXP int CTIAPI CTILink(char*sid1,char*sid2,int type);
//断开通话操作
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
//说明:该动作将会话声音断开,连接到没有声音的电路上,从而听不见任何声音,但是别人之前假如跟它在通话,还需要对别人也下一次这个操作.两个人才真正互相不通
DLLAPIEXP int CTIAPI CTIUnLink(char*sessionId);
//----------媒体操作类 API----------
//播音取按键
//返回:成功:返回 0.其它非零:错误.返回:-11 语音文件名称错误.
//参数:sessionId:会话 id
//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.pcm 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

```
//EndDtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件
//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件
//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件
//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件
//dtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"
//说明:假如只有播音,不收取按键 EndDtmf="",MaxDtmf=0
DLLAPIEXP int CTIAPI CTIPlayDtmf(char*sessionId,char*filename,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*dtmf);
//录音取按键
//返回:成功:返回 0.其它非零:错误.返回:-11 语音文件名称错误.
//参数:sessionId:会话 id
//filename:录音语音文件名称
//EndDtmf:按键结束条件
//MaxDtmf:最大按键个数结束条件
//MaxTimer:按键最大时间结束条件
//TwoDtmfTimer:两个按键间隔结束条件
//dtmf:收到的用户的按键(输出参数)
//说明:按键条件具体参考播音取按键接口
DLLAPIEXP int CTIAPI CTIRecordDtmf(char*sessionId,char*filename,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*dtmf);
//停止播音或者录音等媒体操作
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
//type:停止类型，1 停止 play ,2 停止 record,3 停止所有，其它，错误
DLLAPIEXP int CTIAPI CTIStop(char*sessionId,int type);
//通话过程中对两个通话双方进行录音
//返回:成功:返回 0.其它非零:错误.返回:-11 会话 id2 错误.-12:录音文件名称错误
//参数:sessionId:会话 id
//sessionId2:会话 id2,通话的另外一个会话 sid
//filename:录音语音文件名称
//说明:只能是通话过程中才能调用这个.调用的 sid 必须先有语音资源才行
////假如没有先申请语音,需先使用使用 CTILink(sid,sid,0)申请语音.
DLLAPIEXP int CTIAPI CTIRecord2Sid(char*sessionId,char*sessionId2,char*filename);


//----------------------------------------------会议操作类 API--------------------------------
//加入会议
//返回:成功:返回 0.其它非零:错误.返回:-11 会议号码错误.:-10 超时 500ms 没有收到回应.可以尝试重复调用,
//参数:sessionId:会话 id
//meetno:会议号码，数字号码串
DLLAPIEXP int CTIAPI CTIAddConf(char*sessionId,char*meetno);
//退出会议
//返回:成功:返回 0.其它非零:错误.返回:-11 会议号码错误.
//参数:sessionId:会话 id
```

```
//meetno:会议号码，数字号码串
DLLAPIEXP int CTIAPI CTIOutConf(char*sessionId,char*meetno);
//会议设置
//返回:成功:返回 0.其它非零:错误.返回:-11 会议号码错误.
//参数:sessionId:会话 id
//meetno:会议号码，数字号码串
//type:设置类型，1 说话，0 静音
DLLAPIEXP int CTIAPI CTISetConf(char*sessionId,char*meetno,int type);
//会议录音
//返回:成功:返回 0.其它非零:错误.返回:-11 会议号码错误.-12:录音文件名称错误
//参数:sessionId:会话 id
//meetno:会议号码，数字号码串
//filename:录音语音文件名称
DLLAPIEXP int CTIAPI CTIRecordConf(char*sessionId,char*meetno,char*filename);
//接收传真
//返回:成功:返回 0.其它非零:错误.返回:-12:录音文件名称错误
//参数:sessionId:会话 id
//filename:传真文件名称。必须是*.tif 的文件名次格式
//type:扩展字段，目前没有使用，默认=0
DLLAPIEXP int CTIAPI CTIRecvFax(char*sessionId,char*filename,int type);

//发送传真
//返回:成功:返回 0.其它非零:错误.返回:-12:录音文件名称错误
//参数:sessionId:会话 id
//filename:传真文件名称。必须是*.tif 的文件名次格式
//type:扩展字段，目前没有使用，默认=0
DLLAPIEXP int CTIAPI CTISendFax(char*sessionId,char*filename,int type);

//----------------------------------------------辅助媒体操作类 API--------------------------------
//文本转换为语音操作
//返回:成功:返回 0.其它非零:错误.返回:-11 文本内容非法错误.-12:文件名称错误,-13:非法调用,该函数只能在回调函数里面被调用
//参数:sessionId:会话 id
//ttstext:文本内容
//filename:产生的语音文件名称(输出参数)
DLLAPIEXP int CTIAPI CTITTSChange(char*sessionId,char*ttstext,char*filename);
//语音识别声纹认证申请释放操作
//返回:成功:返回 0.其它非零:错误.-13:非法调用,该函数只能在回调函数里面被调用
//参数:sessionId:会话 id
//cmd: 申请 alloc>0,释放 free=0, 停止 abort <0
DLLAPIEXP int CTIAPI CTIAsrAllocFreeAbort(char*sessionId,int cmd);
//声纹认证操作
//返回:成功:返回 0.其它非零:错误.1:声纹确认不是本人结果,-11:启动识别错误,-12:可能没有申请识别资源,-13:非法调用,该函数只能在回调函数里面被调用
```

```
//参数:sessionId:会话 id
//speakermode:声纹模型名称
DLLAPIEXP int CTIAPI CTISpeakerVerify(char*sessionId,char*speakermode);
//声纹识别操作
//返回:成功:返回 0.其它非零:错误.1:没有识别出结果,-11:启动识别错误,-12:可能没有申请识别资源,-13:非法调用,该函数只能在回调函数里面被调用
//参数:sessionId:会话 id
//speakermodepath:声纹模型目录
//Identifymode:识别出来的声纹模型|录音文件名称(输出参数)
//score:识别的分数(输出参数)
//说明:假如没有识别出结果:Identifymode=|录音文件名称
DLLAPIEXP int CTIAPI CTISpeakerIdentify(char*sessionId,char*speakermodepath,char*Identifymode,int*score);
//语音识别操作
//返回:成功:返回 0.按键输入也是返回 0，但是 score=101，识别没有结果：返回 1，其它非零:错误.返回:-11:启动识别错误,-12:可能没有申请识别资源,-13:非法调用,该函数只能在回调函数里面被调用,-14:语法文件非法.-15:语音文件名称错误.
//参数:sessionId:会话 id
//filename:播放的语音文件名称,
//grammar:语音识别的语法内容,格式:语法类型 语音日志文件，语法类型 3-10 代表 3 个字符串到 10 个字符串。0 代表列表识别:刘德华;张学友
//outbuff:识别出来的文本(输出参数)，比如数字串：1234
//score:识别的分数 0-100(输出参数)
DLLAPIEXP int CTIAPI CTIPlayAsr(char*sessionId,char*filename,char*grammar,char*outbuff,int*score);
//----------------------------------------------辅助控制媒体操作类 API-------------------------------
//获取呼叫类型
//返回:呼叫类型,0 没有呼叫，1，拨入，>1 拨出，<0 错误
//参数:sessionId:会话 id
DLLAPIEXP int CTIAPI CTIGetCallType(char*sessionId);

//获取呼叫信息
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
//caller:主叫号码(输出参数)
//called:被叫号码(输出参数)
//callop:源被叫号码(输出参数)
DLLAPIEXP int CTIAPI CTIGetCallInfo(char*sessionId,char*caller,char*called,char*callop);
//-----------------------------------------------用户消息类 API-------------------------------
//发送用户消息
//返回:成功:返回 0.其它非零:错误.
//参数:sessionId:会话 id
//参数:evt:事件类型
//参数:data:事件数据
//参数:msg:事件附加消息
//说明:为了区别于用户挂机 evt 不要等于-99.
DLLAPIEXP int   CTIAPI CTISendMsg(char* sessionId,int evt,int data,char*msg);
```

```
//等待用户消息
//返回:成功:返回 0.其它非零:错误.-13:非法调用,该函数只能在回调函数里面被调用
//参数:sessionId:会话 id
//参数:evt:事件类型(输出参数)
//参数:data:事件数据(输出参数)
//参数:msg:事件附加消息(输出参数)
//说明:返回-99 表示收到用户挂机信号,返回 99 表示内线用户拍插簧事件.
DLLAPIEXP int CTIAPI CTIWaitMsg(char* sessionId,int *evt,int *data,char*msg);
//-----------------------------------------------用户工具类 API---------------------------------
//获取内线和 VOIP 线路配置信息
//                                                    返                                                    回
user_line="%04d,%04d,%04d,%04d,"==>local_user_start_out,local_user_end_out,local_voip_start_out,local_voip_end_out
DLLAPIEXP int CTIAPI CTIGetLineConfig(char*user_line);
//获取线路状态信息
//返回 0=空闲,1=忙,其它<0  错误
DLLAPIEXP int CTIAPI CTIGetLineState(char*sid);
#ifdef __cplusplus
}
#endif
//---------------------------------------------------------------------------
#endif
//---------------------------------------------------------------------------
```

204. CTI 平台开发 DLL 接口如何使用？

程序员都喜欢看代码，直接跟 C 沟通最快，下面是一个简单的 demo 里面的流程。
大家可以看到，写一个简单的例子，包括注释，代码也就 150 行再加上单线程，和媒体的同步操作，
开发 IVR 流程变成不是那么白色恐怖的事情。

```
int CTIAPI OnRingConnect(char*sid,int ring_connect)
{
 char tmp[1024];
 char dtmf[128],caller[128],called[128],callop[128];
 int res,count=0,calltype;
 if(ring_connect==2)//拨出对方振铃 ring
 {
        calltype=CTIGetCallType(sid);
        printf("user call out CTISwitch sid=%s\n",sid);
        return 0;
 }
 if(ring_connect==3)////拨出对方接通 call out connect...
 {
        CTIGetCallInfo(sid,caller,called,callop);
        printf("out user call conn,caller=%s,called=%s,sid=%s\n",caller,called,sid);
```

```
        calltype=CTIGetCallType(sid);
        printf("ring_connect user call out CTISwitch sid=%s\n",sid);
        CTILink(sid,gsidin,0);
        CTIWaitMsg(sid,0,0,NULL);          //通话.等待挂机
        CTIDropCall(sid);
        return 0;
 }
 if(ring_connect==4)////呼叫挂机 call disconnect...
 {
        return 0;
 }
//拨入流程
 CTIGetCallInfo(sid,caller,called,callop);
 printf("user call in,caller=%s,called=%s,sid=%s\n",caller,called,sid);
 res=CTIAnswer(sid);
 if(res){CTIDropCall(sid);return res;}

 //connect...
 CTIGetCallInfo(sid,caller,called,callop);
 printf("user call conn,caller=%s,called=%s,sid=%s\n",caller,called,sid);

 calltype=CTIGetCallType(sid);
 dtmf[0]=0;
 res=CTIPlayDtmf(sid,"hello","",0,0,0,dtmf);//播放欢迎词
 if(res){CTIDropCall(sid);return res;}

 while(1)
 {
        dtmf[0]=0;
//播放：请输入分机号码按#号确认，查号请按 0，留言请按星号，参加会议请按#号
        res=CTIPlayDtmf(sid,"menu","#*0",10,20,0,dtmf);
        if(res){CTIDropCall(sid);return res;}
        printf("get dtmf=[%s],time=%d\n",dtmf,time(NULL));
        if(dtmf[0] =='\0')
        {
                if(count++>3)
                {
                        res=CTIPlayDtmf(sid,"errortoomany;hangup","",0,0,0,dtmf);
                        CTIDropCall(sid);return res;
                }
                else
                        res=CTIPlayDtmf(sid,"err","",0,0,0,dtmf);
                if(res){CTIDropCall(sid);return res;}
```

```
            continue;
        }
        else
            break;
    }
        if(dtmf[0]=='0')
    {

        res=CTIMakeCall(tmp,"1009","1000","",2);
        if(res){CTIDropCall(sid);return res;}
        strcpy(sidout,tmp);
        printf("makecall ok out sid=%s\n",sidout);
        strcpy(gsidin,sid);
        CTIWaitMsg(sid,0,0,NULL);
        CTIDropCall(sid);
        return res;
    }
    if(dtmf[0]=='1')
    {
        res=CTIPlayDtmf(sid,"bye","",0,0,0,dtmf);
        if(res){CTIDropCall(sid);return res;}
        CTIDropCall(sid);
        return res;
    }
    if(dtmf[0]=='*')
    {           //播放：请在听到滴声之后留言，结束留言请按#号
        res=CTIPlayDtmf(sid,"startrecord;beep","",0,0,0,dtmf);
        if(res){CTIDropCall(sid);return res;}
        res=CTIRecordDtmf(sid,"system/ad1rec2","",1,120,0,dtmf);
        if(res){CTIDropCall(sid);return res;}
    //播放：用户的留言，然后播放谢谢使用再见
        res=CTIPlayDtmf(sid,"system/ad1rec2;bye","",0,0,0,dtmf);
        if(res){CTIDropCall(sid);return res;}
        CTIDropCall(sid);
        return res;
    }
    if(dtmf[0]=='#')
    {
        res=CTIAddConf(sid,"1000");
        if(res){CTIDropCall(sid);return res;}
        CTIWaitMsg(sid,0,0,NULL);
        CTIDropCall(sid);
        return 0;
    }
```

```
 tmp[0]=0;
//播放：静音，等待输入后续的分机号码
 res=CTIPlayDtmf(sid,"m02","#",15,15,0,tmp);
 if(res){CTIDropCall(sid);return res;}
 printf("get dtmf=[%s],time=%d\n",tmp,time(NULL));
 strtok(tmp,"#");
 strcat(dtmf,tmp);

 res=CTIMakeCall(tmp,"1000",dtmf,"",3);
 if(res){CTIDropCall(sid);return res;}
 strcpy(sidout,tmp);
 printf("makecall externion    ok out sid=%s.called=%s",sidout,dtmf);
 strcpy(gsidin,sid);

 CTIWaitMsg(sid,0,0,NULL);
 CTIDropCall(sid);
 CTIDropCall(sidout);
 return 0;
}
int main()
{
 char sid[128],config[128];
 int res;
 res= Init_Cti();
 printf("Init_Cti() res=%d\n",res);
 if(res) return 0;
 strcpy(config,"./cti.conf");
 res = CTIOpenIvr(config,OnRingConnect);
 int Stop=0;
  while (1)
 {
        char ch=getch();
        if(ch>='0' && ch<='9')
        {
              sprintf(sid,"%d",ch-'0');
              CTIDropCall(sid);
        }
        if(ch=='m') res=CTIMakeCall(sid,"1001,"1002","",2);
        if(ch=='q') break;
        Sleep(1000);
        if(Stop) break;
      }
 CTICloseIvr();
 printf("main end\n");
```

```
 return 0;
}
```

205. CTI 平台 Flash 插件接口有哪些功能？

CTI 平台的 Flash 接口提供完备的呼叫中心坐席控制功能。
可以完全在插件上进行操作，也可以通过 js 的接口进行操作。
这样其它 CRM 系统想要集成呼叫中心的功能只要基于这个 js 的接口进行二次开发就可以，根据已经有的集成经验，一般的坐席功能，一个 web 工程师 5 工作日内即可以完成集成开发。
js 的的具体接口以及说明如下：
1.接口插件使用 flash 插件方式
接口可以完全在插件上进行操作，也可以通过 js 的接口进行操作。
使用 js 的接口使不熟悉的用户集成呼叫中心成为轻而易举的事情。
2．提供来电回调 js 的函数
通过回调 js 函数，可获取到用户来电号码和通话记录表里面的 cdrid 等参数
使用这些信息可以获取到用户的 360 度信息。
3．坐席常用函数具体说明：
3.0 连接
loginRed5(red5url:String)
参数:red5url:服务器连接串。格式：rtmp://ip:port
成功：返回 0；其他错误代码

3.1 签入
坐席上班的时候要调用签入函数，或者使用插件上的签入按键进行签入。
checkIn(hotline:String,seatno:String,password:String,station:String,stationtype:int,checkintype:int)
参数：hotline 企业热线号码，没有填""
seatno 坐席号码
password：密码
station：绑定的分机号码或者手机号码
stationtype:绑定电话类型
checkintype:签入类型
成功：返回 0；其他错误代码
其他说明：hotline 不是唯一所以可以填也可以填""，
seatno 是数据库里面的 workno，是全局 唯一。
stationtype 绑定电话类型：0：pstn 电话坐席，1：sip，IP 坐席
checkintype 签入类型：
checkintype=0 web 登陆的签入，1 非 web 登陆的签入（后台签入），一般自己签入都是 0
checkintype=1 是后台签入，一般在坐席管理界面用帮其他座席设置,也可以是管理坐席自己操作坐席表
3.2 签出
坐席要下班的时候调用此函数，此函数在签入成功之后才能调用
checkOut(seatno:String)
参数：seatno 坐席号码
成功：返回 0；其他错误代码

3.3 呼叫

坐席要呼叫用户的时候调用此函数，此函数在签入成功之后才能调用

makeCall(seatno:String,calledtelno:String,externtion:String)

参数：seatno 坐席号码

calledtelno 被叫号码，非本地的外地固话或者短号码要加区号

externtion 扩展的信息

成功：返回 0；其他错误代码

3.3 挂机

坐席要挂掉正在通话的用户的时候调用此函数，此函数在签入成功之后才能调用

dropCall(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误代码

3.4 设置坐席忙闲状态

坐席要暂时离开的时候调用此函数，此函数在签入成功之后才能调用

SetSeatState(seatno:String,seatstate:int)

参数：seatno 坐席号码

seatstate 坐席状态，=0 设置空闲，=1 设置忙

成功：返回 0；其他错误代码

3.5 来电和接口函数调用回调 js 的函数

onCallinCallouRing(callercalledinfo)

说明：

有用户来电，或者坐席拨出之后用户振铃，系统会调用这个函数

用户拨入：callercalledinfo 内容格式： "callin:caller="+callinfoH.value.caller+"called="+callinfoH.value.called;

比如用户拨入 callin:caller=13606060253called=3010110uid=xxxxcdrid=xxxx

或者坐席拨出 callou:caller=3010110called=13606060253uid=xxxxcdrid=xxxxext=xxx

或者 操作的回调： callop:opfun=XXXXresult=X

说明：result=0 ok result<0 错误代码

3.6 播音函数

playnumvox(numberstring:String)

参数：numberstring 要播放的语音：'0'-'9' 'r'

'r' 表示振铃声音

成功：返回 0；其他错误代码

3.7 下班不签出，后台值班

坐席要下班了，但是还考虑使用手机接客户的电话进行后台电话值班的时候调用此函数，此函数在签入成功之后才能调用

logoutOnDuty(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误代码

3.8 查询取席的各种状态

GetSeatCallState(seatno:String)

参数：seatno 坐席号码

返回: -1:未连接到 Red5(或者连接到 Red5 失败),0:签出(或者签入失败),1:签入,
2:上线,3:服务中与用户通话,4:置忙,5:保持,6:咨询坐席,7:咨询外线,
8:三方通话,9:监听坐席,10:插话,11 :上线置忙,
12：坐席拨打用户，用户正在振铃。

4．坐席高级函数具体说明：

4.1 坐席上线

坐席要接通电话到系统的时候调用此函数，此函数在签入成功之后才能调用

online(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：上线之后，系统会呼叫坐席，坐席绑定的电话或者 ip 电话跟系统是接通的。

4.2 坐席下线

坐席要挂掉自己绑定的电话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，或者上线之后才能调用

offline(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：下线之后，系统挂断坐席，假如坐席跟用户通话，用户也会被挂机

4.3 保持用户

坐席要让用户听音乐，自己找一些资料的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中才能调用

holduser(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：保持之后，系统给用户播放音乐。

4.4 结束保持

坐席在保持用户之后，要重新跟用户通话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，保持用户（holduser）之后才能调用

unholduser(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：结束保持之后，坐席重新跟用户对话

4.5 咨询其他坐席

坐席要咨询其他坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中才能调用

callSeat(seatno:String,toseatno:String)

参数：seatno 坐席号码

toseatno 被叫坐席号码

成功：返回 0；其他错误

4.6 关闭正在咨询的坐席

坐席要关闭正在咨询的坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，咨询其他坐席（callSeat）之后才能调用

freeseat(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：关闭咨询，可以是自己咨询别的坐席，也可以是被别的坐席咨询

4.7 转接正在咨询的坐席

坐席要转接正在咨询的坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，咨询其他坐席（callSeat）之后才能调用

transSeat(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

4.8 咨询其他外线用户

坐席要咨询其他外线用户的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中才能调用

calltouser(seatno:String,calledtelno:String,externtion:String)

参数：seatno 坐席号码

calledtelno 被叫号码，非本地的外地固话或者短号码要加区号

externtion 扩展的信息

成功：返回 0；其他错误

4.9 挂掉正在咨询的外线用户

坐席要挂掉正在咨询的外线用户的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，咨询其他外线用户（calltouser）之后才能调用

droptouser(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

4.10 转接正在咨询的外线用户

坐席要转接正在咨询的外线用户的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，咨询其他外线用户（calltouser）之后才能调用

transCall(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

4.11 三方通话

坐席要进行 3 方通话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，咨询其他外线用户（calltouser）之后才能调用

inmeet(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

4.12 结束三方通话

坐席要结束 3 方通话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，三方通话（inmeet）之后才能调用，

outmeet(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：outmeet 之后 对方两个人在继续通话

5. 质检班长坐席高级控制函数具体说明：

5.1 开始监听其他坐席

班长坐席要监听其他坐席的时候调用此函数，此函数在签入成功之后才能调用

moniter(seatno:String,toseatno:String)

参数：seatno 坐席号码

toseatno 被叫坐席号码，注意：不是坐席号码

成功：返回 0；其他错误

5.2 结束监听其他坐席

班长坐席要结束监听其他坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，开始监听其他坐席（moniter）之后才能调用，

outmoniter(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

5.3 插话

班长坐席在监听之后要插话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，开始监听其他坐席（moniter）之后才能调用，

insert(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

5.4 结束插话

班长坐席在监听插话之后要停止插话的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，开始插话其他坐席（insert）之后才能调用，

outinsert(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

5.5 强接其他坐席

班长坐席在监听的时候要强接其他坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，开始监听其他坐席（moniter）之后才能调用，

robcall(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：强接之后变成由班长给用户服务

5.6 强拆其他坐席

班长坐席在监听的时候要强拆其他坐席的时候调用此函数，此函数在签入成功之后才能调用

此函数要在通话中，开始监听其他坐席（moniter）之后才能调用，

stopcall(seatno:String)

参数：seatno 坐席号码

成功：返回 0；其他错误

说明：强拆之后变成坐席和用户服务结束

5.7 签入其他坐席

签入其他坐席上班的时候要调用此签入函数

tocheckIn(hotline:String,seatno:String,password:String,station:String,toseatno:String,tostationtype:int,tocheckintype:int)

参数：hotline 企业热线号码，没有填""

seatno 坐席号码

password：被签入的坐席的密码

station：被签入的的坐席绑定的分机号码或者手机号码

toseatno:被签入的坐席号码

tostationtype:绑定电话类型

tocheckintype:签入类型

成功：返回 0；其他错误代码

5.8 签出其他坐席

签入其他坐席上班的时候要调用此签入函数

tocheckOut(seatno:String,toseatno:String)

参数：

seatno 坐席号码

toseatno:被签入的坐席号码

成功：返回 0；其他错误代码

6.坐席的状态变迁

坐席签入状态下：

-》签出

-》置忙、置闲

-》上线、下线

-》通话(主叫、被叫)

坐席通话中：

-》挂机

-》保持 -》 接回

-》咨询坐席

-》 结束咨询（可以再马上咨询其他坐席，需要点取消保持才能恢复跟用户通话）

-》 转接坐席

-》咨询外线

-》 结束咨询（可以再马上咨询其他用户，需要点取消保持才能恢复跟用户通话）

-》 转接外线

-》 三方通话（用户，自己和被咨询的用户进入三方通话）

-》结束三方通话(自己挂机，用户个和咨询的用户通话)

班长控制(签入状态下(需要班长先上线))：

-》监听坐席 -》 结束监听

-》 插话 -》结束插话

-》 强接

-》 强拆

7. 错误代码说明：

返回 0 表示成功

非零错误代码：

-19001//坐席号码未签入

-19102//坐席状态错误

-19004//数据库执行错误

-19005//数据库无结果集

-19006//数据库结果集错误

-19010//签入密码错误

-19011//坐席被关闭

-19012//坐席台号为空

-19013//用户号码为空

-19014//转接用户号码为空

-19021//非坐席管理员权限

-19100//呼叫操作错误

----------状态类----------

-19200//坐席已经被占用

-19201//没有用户呼叫

-19202//坐席已经是保持状态
-19203//坐席不在服务用户
-19204//坐席未上线
-19205//坐席已经呼叫
-19206//坐席没有保持
-19207//坐席已经呼叫了转接用户
-19208//无转接用户呼叫会话
-19209//无转接用户呼叫记录
-19210//坐席还未和转接用户对话
-19211//坐席还未和用户对话
-19212//坐席已经在会议中
-19213//坐席不在会议中
-19214//坐席正在咨询坐席
-19215//被咨询的坐席已经分配服务
-19216//被咨询的坐席已经置忙
-19217//被咨询的坐席没有签入
-19218//没有咨询坐席操作
-19219//被咨询的坐席没有进入服务
-19220//被咨询的坐席没有发起呼叫
-19221//被咨询的坐席没有接通
-------客户端类----------
-10//业务软件没有连接上控制条
-98//业务软件和控制条通信错误
-99//控制条和服务器通信错误
-100//坐席号和控制条不一致
-101//控制条还没有签入
-20001//控制条没有初始化 dll
-20002//控制条没有连接上服务器
-20003//控制条没有登陆上服务器
-20004//控制条没有执行过签入动作

206. CTI 平台 DLL 实网环境中如何跟踪和解决崩溃？

非开发环境的实网环境下的 debug

a.使用微软的工具

可以使用微软的安装包,也可以使用安装之后的，免安装版本：

b.需要提供开发环境编译产生的所有工程的 pdb 文件

以 test.ext 为例：非开发环境下的 debug 步骤：

.0 准备提供 test.pdb 在 test.exe，Pdb 文件名次要和编译产生的 exe 名次一样，不能修改。Releae 版本也可参数 pdb 文件。

.1 免安装版本：Windows_debug.rar 解压都某个目录，Windows_debug.rar 可以到 附录的 ftp 上匿名下载

.2 命令行下执行： “Windows Kits\8.0\Debuggers\x86\adplus_old.vbs” -crash -pn test.exe -o "c:\debug"

Debug 目录是预先建立好的输出 debug 日志的目录

.3 一旦程序崩溃会参数 dump 和 log 文件

比如下面的 code

```
int check_time()
{
  static char *p;
  static int c=1;
        printf("time=%d\n",time(NULL));
        if((c++)%10==0)
        {
              printf("time=%d,%s\n",time(NULL),time(NULL));
              *p=1;
              p++;
        }
        return 0;
}
```

崩溃之后,打开 PID-4324__TEST.EXE__Date_11-14-2012__Time_10-11-3737.log 文件可以找到是哪个函数里面崩溃，这样对崩溃的跟踪就比较准确。

```
Faulting stack below ---
 # ChildEBP RetAddr  Args to Child
00 0012fe5c 004022cb 0042aa80 0042818e 0012fe9c test!_output+0x5b3
01 0012fe88 00401ebe 00428184 50a2fde1 50a2fde1 test!printf+0x5b
02 0012fee8 00401f82 00000000 00000000 7ffd4000 test!check_time+0x6e
03 0012ff48 004036a9 00000001 006c0f88 006c0fd0 test!main+0x62
04 0012ff88 77741174 7ffd4000 0012ffd4 77d0b3f5 test!mainCRTStartup+0xe9
WARNING: Stack unwind information not available. Following frames may be wrong.
05 0012ff94 77d0b3f5 7ffd4000 73482437 00000000 kernel32!BaseThreadInitThunk+0x12
06 0012ffd4 77d0b3c8 004035c0 7ffd4000 00000000 ntdll!RtlInitializeExceptionChain+0x63
07 0012ffec 00000000 004035c0 7ffd4000 00000000 ntdll!RtlInitializeExceptionChain+0x36
```

第十二章 FreeSwitch 接口卡 Sangoma 部分

207. FreeSwitch 部署为啥需要支持接口板卡？

假如你是通过其他的 VOIP 落地提供商来提供出口落地的。那是你的系统不需要接口板卡，但是这个通过其他的 VOIP 落地提供落地方案作为系统运营是有风险的。由于众所周知的原因，其他的 VOIP 落地提供商说不定哪天就被灭了，所以最好还是使用自己的落地出口。

搞过电信行业的人都知道，之前跟运营商的接口都是 E1 接口，所以支持 E1 接口板卡是必须的，不仅仅是 FreeSwitch 支持，其他的比如 dialogic 的 HMP，很早以前 HMP 也是不支持 E1 接口卡的，后面 HMP3.0 之后也是同样提供 DNI 的接口卡的支持。通过连接到运营商的交换系统上。 最近几年运营商开始推广 IMS 系统，但是在大多城市数地方 E1 接口接入还是必须的支持的。还没听说哪个运营商不支持 E1 接口的。当然 E1 的接口你也可以使用 VOIP 网关作为落地出口。但是无论从性价比，延时，运维等各个方面来看，使用 FreeSwitch 自己集成支持的板卡都是最好的 E1 落地方案，没有之一。

不信请看，有一次促销，最常用的 1-8E1 的板卡型号和价格如下：

SKU		MSRP	Promo Price
101DE 1 T1/E1/J1		~~$1,000~~	$250
102DE 2 T1/E1/J1		~~$1,600~~	$400
104DE 4 T1/E1/J1		~~$2,700~~	$675
108DE 8 T1/E1/J1		~~$4,000~~	$1,000

8E1 1000 美刀相当于每个 E1 1000 RMB。没有听说过有比这个便宜的 VOIP 网关之类的东东。要比这个便宜除非你自己使用交换芯片设计生产。 **J**

208. FreeSwitch 支持哪些接口板卡？

FreeSwitch 接口板卡模块叫做 FreeTDM，具体参见：

http://wiki.freeswitch.org/wiki/FreeTDM#Configuration

这个上面写得很多，看起来很晕。归纳起来就是首先支持 Digium 和 Sangoma 两个厂家的板卡。

然后 Sangoma 板卡再细分就有模拟卡，数字 E1 接口卡，转码卡等等。

再然后 Sangoma 板卡信令模式有： ISDN Modules， SS7 Modules， Analog， MFC-R2

各种信令不一一介绍。

209. 为啥选择 Sangoma 接口卡？

FreeSwitch 接口板卡模块 FreeTDM 我选择 Sangoma 板卡作为接口卡。

理由：

1. FreeTDM 模块是 Sangoma 公司开发维护升级的。

2. Sangoma 接口卡同时支持 LINUX 和 Windows。这个适合产品部署和选型。有些东东可能是一个产品，而不是一个系统，那么就只能使用一台机器，这个时候，可以选择 LINUX 或者 Windows 其中某个操作系统是很关键。所以在我看来同时支持 Windows 和 Linux 是必须的。将来系统产品部署会有很大的选择余地。

3. Sangoma 渠道比较多，当然板卡的稳定性也不错，价格也还行。

210. Sangoma 数字接口卡具体型号有哪些？

Sangoma E1 板卡从 1E1 到 16E1 都有。带回声，不带回声都有，带回声的由于有 DSP（DSP 芯片 Sangoma 当然也是采购的，不会是自己生产的）所以 DSP 成本是很高的，相对没有 DSP 的板卡贵了差不多 1 倍。我们一般的应用，使用选择不带 DSP 芯片的接口卡就可以。毕竟做这种开源的集成系统，无论老板员工还是客户大多数都是屌丝，不是高富帅,对价格很敏感，能省还是省省吧。:）

Sangoma E1 板卡卡插口有 PCI 和 PCI-E 两种，这两种是不兼容的，目前的 PC server 通常都支持 PCI-E。PCI 属于老式的接口，在我看来已经属于 OUT 类型的了，我建议采购的时候要采购 PCI-E 的插口的板卡。

下图是 PCI 插口的 8E1 的接口 E1 卡，注意看挡板比较高，

下图是 PCI-E 插口的 8E1 的接口 E1 卡,跟上面的对比区别在于支持的插槽不一样。

下图是 PCI-E 插口的 16 E1 的接口 E1 卡

这个卡需要专门的扩展线和接头（需要额外花钱）才能接入 E1 线路。
总代理提供的具体型号列表如下：

产品订货代码	产品	价格
A101E	A101 PCI-E 1 口不带回声模块	
A101D	A101 PCI 1 口带回声模块	
A101DE	A101 PCI-E 1 口带回声模块	
A101	A101 PCI 1 口不带回声	
A102	A102 PCI 2 口不带回声	
A102E	A102 PCI-E 2 不带回声	
A102D	A102 PCI 2 口带回声	
A102DE	A102 PCI-E 带回声	

A104	A104 PCI 4 口不带回声	
A104E	A104 PCI-E 4 口不带回声	
A104D	A104 PCI 4 口带回声	
A104DE	A104 PCI-E 4 口带回声	
A108	A108 PCI 8 口不带回声	
A108E	A108 PCI-E 8 口不带回声	
A108D	A108 PCI 8 口带回声	
A108DE	A108 PCI-E 8 口带回声	
A116E	16E1,支持 PCI-E，无回声 DSP	
A116DE	16E1，支持 PCI-E，带回声 DSP	
SPEC-A116-PNLKIT	机架挡板+连接线（A116 必配）	

其中带 D 的是有带 DSP 的也就是带回声消除功能的，带 E 的是 PCI-E 的插口
注意 16E1 的 A116E 和 A116DE 需要额外花钱买连接线和机架挡板。

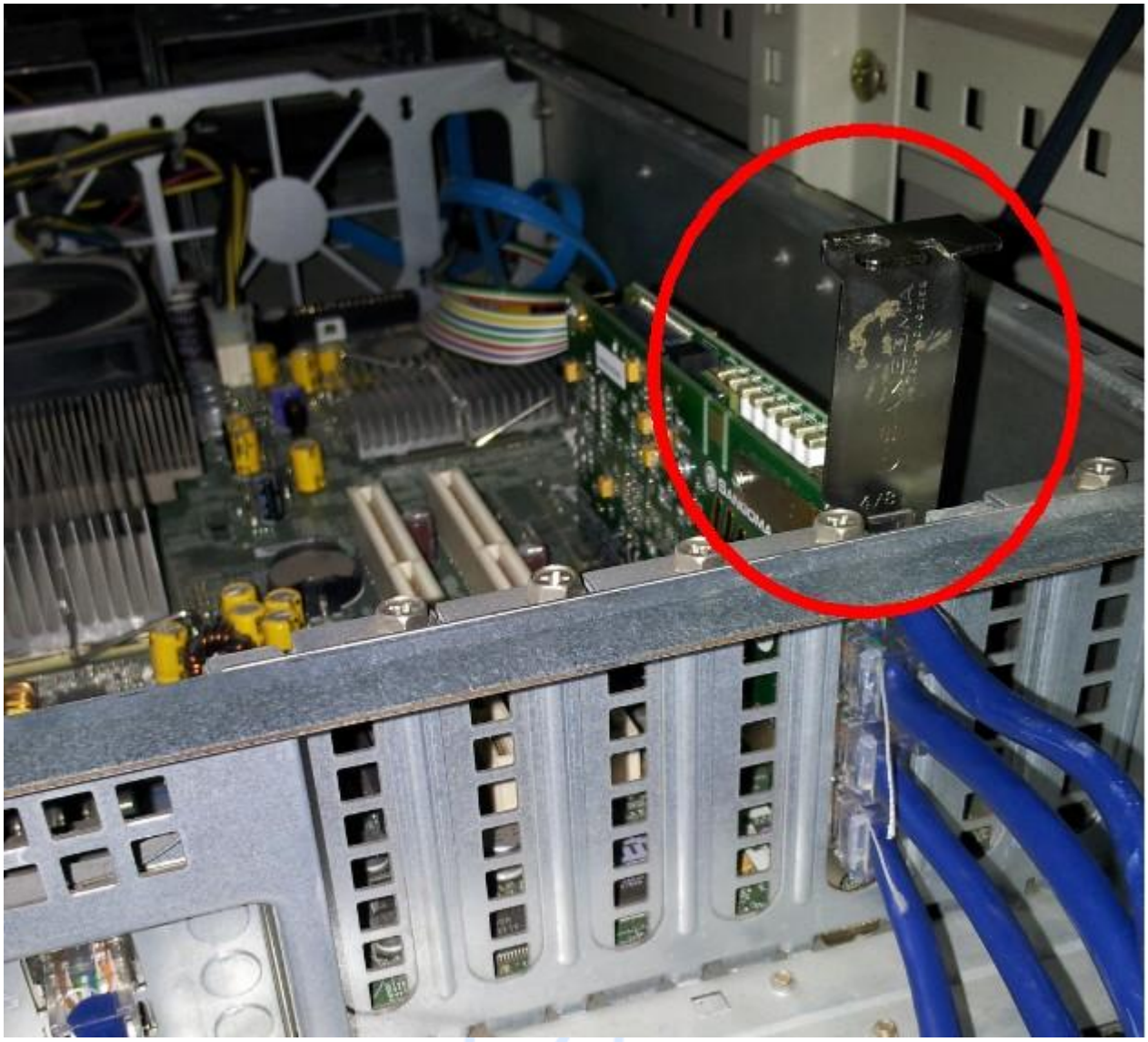
211. Sangoma 板卡硬件安装需求有哪些？

Sangoma 板卡是 PCI 或者 PCI-E （X1 长）接口的卡，上面我说到 PCI 已经 OUT 了我建议采购的时候要采购 PCI-E 的插口的板卡。

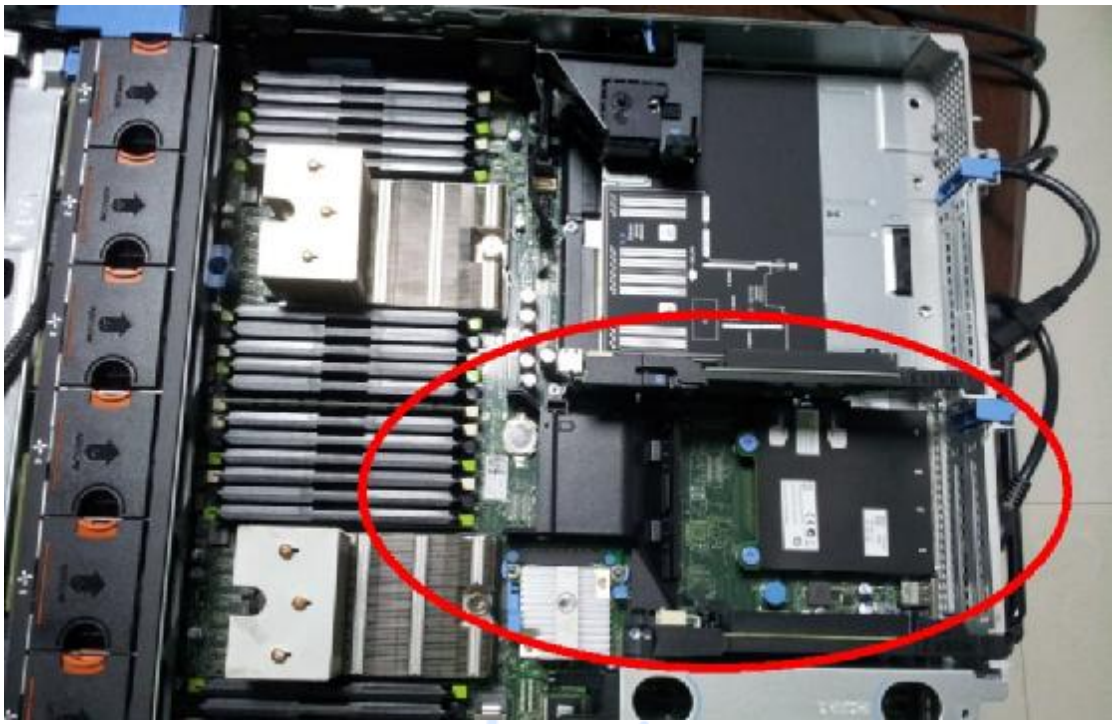
因此第一个需求是需要 PCI-E 的插槽（棕色）才能插，目前的 PC Server 和 PC 机器通常都有提供 PCI-E 的插槽。假如是使用工控机的时候要注意选有 PCI-E 的插槽。

Sangoma 板卡长度比较短，大约 10 多厘米，但是板卡的挡板比较高，超过了 2U，普通的 2U 的 PC 服务器竖插是插不上去的。

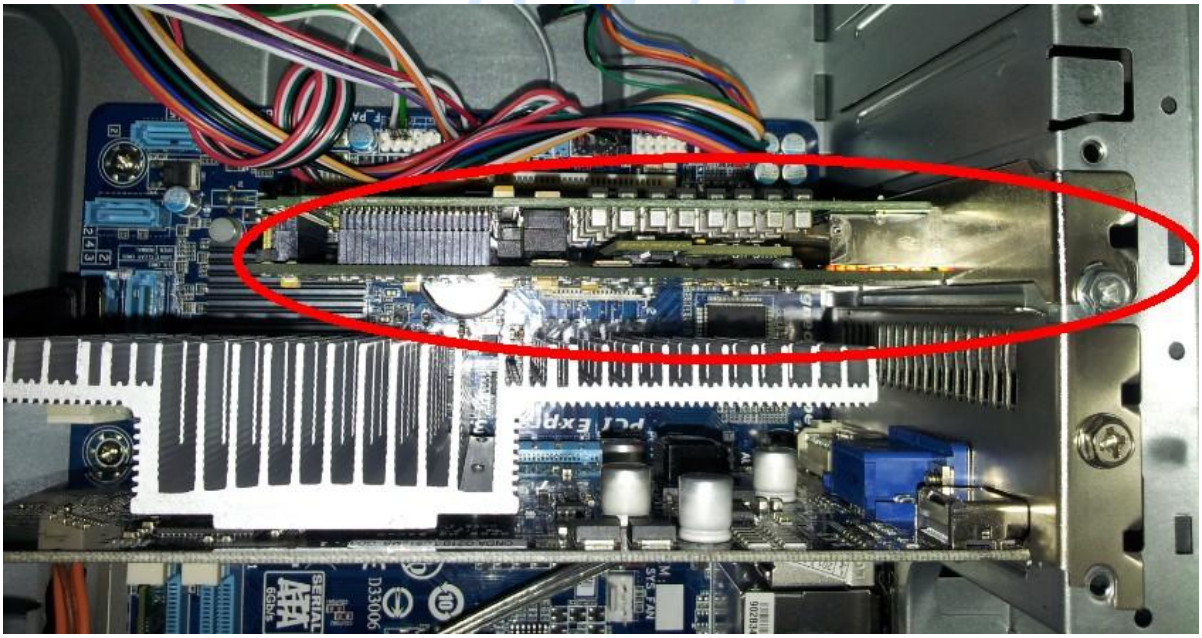
准确说应该是插上去之后无法上螺丝。如下图：



因此假如是 2U 的 PC 服务器，需要能支持横插 PCI-E 的接口。比如 dell R720 服务器，是可以横插上去的。如下图的插槽位置：



买卡之前最好先落实清楚，能否插进去。
或者可以考虑使用工控机，或者使用普通 PC 也可以插上去
如下图：



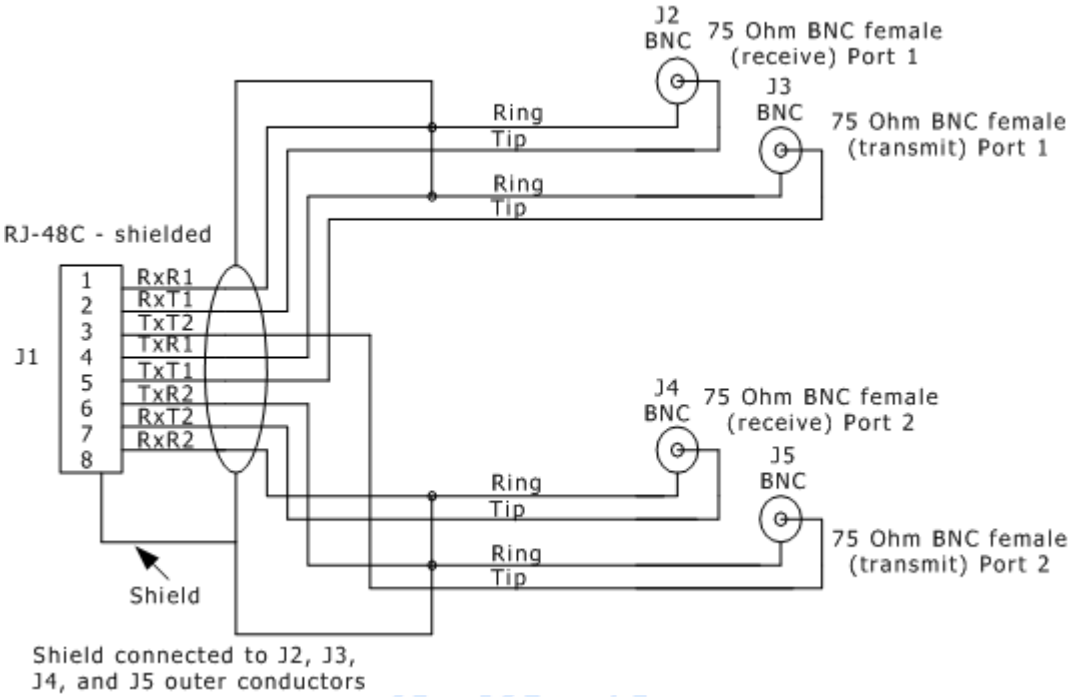
我建议还是使用 2U 的 PC Server,一来可靠性比较高，二来 CPU 可以选择的型号很多，也可以选择多个 CPU。毕竟 FreeSwtich 是完全靠 CPU 做语音处理，特别是在有语音编码转换的使用场景下，工控机和普通 PC 机器的 CPU 性能不行容量自然上不去。

还有一个要注意的是 8E1 的卡只有 4 个口（RJ48/RJ45 的口），一个口是两个 E1，
具体排列如下：
第 1　5　E1
第 2　6　E1

第 3　7　E1

第 4　8　E1

由于一个口两个 E1，因此 E1 的线序需要了解一下，下图是使用 75 欧 E1 的转接到接口的做线方法：

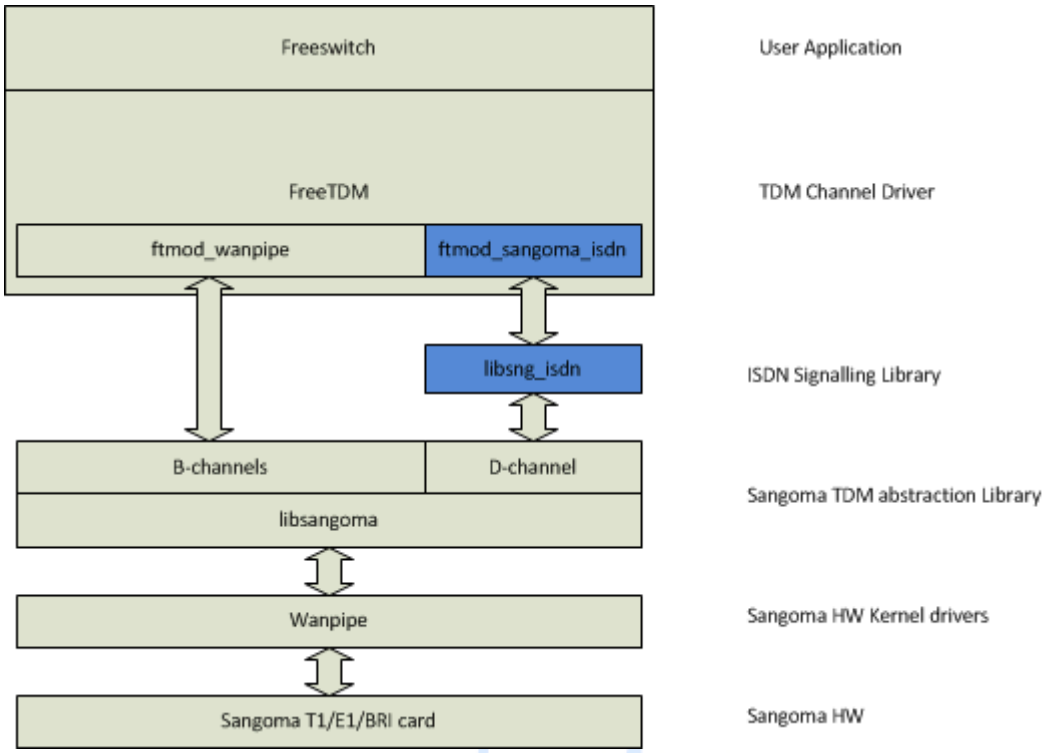


212.　Sangoma 板卡使用哪种模式？

Sangoma 板卡 使用模式有 ftmod_zt，ftmod_pika，和 ftmod_wanpip 模式

Sangoma 建议在 FreeSwtich 里面使用 ftmod_wanpipe 模式，也就是 Wanpipe mode,就是说需要安装 Sangoma 的 wanpipe 驱动，然后配置使用。

这个 ftmod_wanpipe 模式下的 Sangoma 卡的 ISDN 协议栈如下：



213. Linux 下 Sangoma 板卡如何安装如何配置 ISDN-PRI 信令？

Sangoma 板卡 linux 下的安装小有复杂，以 Centos 6.4 64 位 为例安装步骤如下：

先确认 Linux 是 32 位还是 64 位：可以使用 getconf LONG_BIT 或者 uname –m 查看：

[root@sipserver log]# getconf LONG_BIT

64

或者

[root@sipserver log]# uname -m

x86_64

x86_64 表示 64 位，假如是 i686 之类的表示 32 位

然后获取安装包：

wanpipe 安装包

wget ftp://ftp.sangoma.com/linux/current_wanpipe/wanpipe-current.tgz

isdn-pri 信令协议栈安装包

32 位 Linux 获取地址:

wget ftp://ftp.sangoma.com/linux/libsng_isdn/libsng_isdn-current.i686.tgz

64 位 Linux 获取地址:

wget ftp://ftp.sangoma.com/linux/libsng_isdn/libsng_isdn-current.x86_64.tgz

由于 sangoma 网站访问巨慢，没有比蜗牛快多少，上面这些驱动安装包可以到 42.120.20.89 上 ftp 获取。

上面的 wanpipe-current.tgz 和 libsng_isdn-current.x86_64.tgz 取到/usr/src 目录下

然后执行：

tar –xzvf wanpipe-current.tgz
tar –xzvf libsng_isdn-current.x86_64.tgz
出来 /usr/src/wanpipe-7.0.5 目录和 /usr/src/libsng_isdn-7.27.2.x86_64 目录
然后就可以开始编译了 wanpipe：
cd /usr/src/wanpipe-7.0.5
make freetdm
然后就安装 wanpipe：
make install
然后安装 libsng_isdn：
cd /usr/src/libsng_isdn-7.27.2.x86_64
make install
然后 下载 freeswitch-1.2.12.tar.bz2，比如放到/home/fs 目录下，
然后
tar –xzvf freeswitch-1.2.12.tar.bz2
出来 fs 的目录：
/home/fs/freeswitch-1.2.12
然后
修改 /home/fs/freeswitch-1.2.12/modules.conf 文件
文件里面最后几行里面的 mod_freetdm 注释去掉：
#../../libs/freetdm/mod_freetdm
改为
../../libs/freetdm/mod_freetdm
然后 执行脚本命令
cd /home/fs/freeswitch-1.2.12/
./bootstrap.sh
执行 配置命令

./configure
默认安装路径是 /usr/local/freeswitch
然后开始编译：
make
然后安装 fs
make install

Ok 之后，开始配置 FreeTDM：
cd /usr/sbin/
使用 wancfg_fs 配置语音板卡和生成 freetdm 的配置文件,理论上也可以拷贝已经配置好的配置文件到相应的目录下就可以
wancfg_fs
启动之后一些主要的配置参数选择如下：
Would you like to change FreeSWITCH Configuration Directory?
Default: /usr/local/freeswitch/conf
1. NO
2. YES

[1-2, ENTER='NO']:
回车
Select media type for AFT-A108DE on port 1 [slot:4 bus:9 span:1]
1. T1
2. E1
3. Unused
4. Exit
选择 2
Configuring port 1 on A108DE as E1, line coding:HDB3, framing:CRC4
1. YES - Keep these settings
2. NO - Configure line coding and framing
选择 2
Select line coding for port 1 on A108DE
1. HDB3
2. AMI
选择 1
Select framing for port 1 on A108DE
1. CRC4
2. NCRC4
3. UNFRAMED
选择 2 无 CRC4 校验，如果运营商有 CRC4 校验选择 1
Select clock for AFT- A108DE on port 1 [slot:4 bus:9 span:1]
1. NORMAL
2. MASTER
选择 1
Select Switchtype for AFT- A108DE on port 1 [slot:4 bus:9 span:1]
1. EuroISDN/ETSI
2. QSIG
选择 1
Select signalling type for AFT- A108DE on port 1 [slot:4 bus:9 span:1]
1. PRI CPE
2. PRI NET
选择 1
Select dialplan context for AFT- A108DE on port 1
1. default
2. public
3. Custom
选择 1
Input the dialing group for this port:
输入 1
Would you like to enable hardware DTMF detection?
1. YES
2. NO
选择 1

Would you like to enable hardware fax detection?
1. YES
2. NO
选择 1
Port 2 on AFT- A108DE configuration complete...
Press any key to continue:

假如你的板卡是多个 E1，比如 8 个 E1 会一直提示你配置，实际上只要配置一个 E1 就可以，后续就保存退出。其他的在后面有说咋手工配置。
配置完成会产生以下文件：
板卡的资源配置文件：
/etc/wanpipe/wanrouter.rc
板卡的端口配置文件 wanpipe:
/etc/wanpipe/wanpipe1.conf
和
FreeSwitch 的 freetdm 配置文件：
/usr/local/freeswitch/conf/freetdm.conf
Freetdm 的 E1 配置文件,主要是对应的拨号计划配置文件：
/usr/local/freeswitch/conf/autoload_configs/freetdm.conf.xml
配置好了 之后
修改拨号计划：

```
<extension name="Local_Extension2">
        <condition field="destination_number" expression="^([0-9]\d+)$">
                <action application="export" data="dialed_extension=$1"/>
               <action application="set" data="call_timeout=30"/>
                <action application="set" data="record_sample_rate=8000"/>
                <action application="export" data="RECORD_STEREO=false"/>
               <action application="set" data="hangup_after_bridge=true"/>
               <action application="set" data="continue_on_fail=true"/>
               <action application="bridge" data="freetdm/wp1/a/$1"/>
          </condition>
        </extension>
```

启动 FreeSwicth。
接上 E1 线路。
然后再使用命令看 E1 的状态
freeswitch@WIN-6DS8EHJLLUC> ftdm sangoma_isdn show_spans
span:wp1 physical:OK signalling:UP
Command executed OK
使用软电话注册上来就可以拨打测试。

214. Sangoma 板卡 LINUX 下如何手工增加 E1 端口配置？

Sangoma 板卡 linux 下手工扩展 E1 配置的步骤如下：

1.手工 增加/usr/local/freeswitch/conf/freetdm.conf 里面的

[span wanpipe wp2]。。。

扩展到 8 个

2.手工 增加 /usr/local/freeswitch/conf/autoload_configs/freetdm.conf.xml 里面的

```
<span name="wp2" cfgprofile="my_pri_te_1">
                        <param name="dialplan" value="XML"/>
                        <param name="context" value="default"/>
                </span>
                <span name="wp3" cfgprofile="my_pri_te_1">
                        <param name="dialplan" value="XML"/>
                        <param name="context" value="default"/>
                </span>
                。。。扩展到 8 个
```

3./etc/wanpipe/wanpipe1.conf 手工增加： wanpipe2.conf ...wanpipe8.conf 扩展到 8 个

4./etc/wanpipe/wanrouter.rc

WAN_DEVICES="wanpipe1 "

增加修改为：

WAN_DEVICES="wanpipe1 wanpipe2 wanpipe3 wanpipe4 wanpipe5 wanpipe6 wanpipe7 wanpipe8 "

5. wanrouter stop

再 wanrouter start

然后查看

```
[root@sipserver dialplan]# wanrouter status

Devices currently active:
        wanpipe1 wanpipe2 wanpipe3 wanpipe4 wanpipe5 wanpipe6 wanpipe7 wanpipe8
....
Wanrouter Status:
Device name | Protocol | Station | Status          |
wanpipe8    | AFT TE1  | N/A     | Connected       |
wanpipe7    | AFT TE1  | N/A     | Connected       |
wanpipe6    | AFT TE1  | N/A     | Connected       |
wanpipe5    | AFT TE1  | N/A     | Connected       |
wanpipe4    | AFT TE1  | N/A     | Connected       |
wanpipe3    | AFT TE1  | N/A     | Connected       |
wanpipe2    | AFT TE1  | N/A     | Connected       |
wanpipe1    | AFT TE1  | N/A     | Connected       |
```

215. Sangoma 板卡 wanpipe.conf 配置文件缺失如何补充？

在 tar 包的源码编译的 FreeSwitch 里面，启动会提示缺少 wanpipe.conf 文件的错误告警，可以自己手工补充这个文件（不补充貌似也可以使用），

wanpipe.conf 文件放在 /usr/local/freeswitch/目录下
内容如下：

```
[defaults]
; User space interval at which data will be delivered
codec_ms => 20

; wink and flash interval
wink_ms => 150
flash_ms => 750

; size of the driver queue of elements of MTU size
; typical case is 10 elements of 80 bytes each (10ms of ulaw/alaw)
; don't mess with this if you don't know what you're doing
; txqueue_size => 10
; rxqueue_size => 10
```

216. Windows 下 Sangoma 如何编译 mod_freetdm.dll 等文件？

Sangoma 开发包在 Windows 下默认是没有 mod_freetdm.dll 的，先编译好 FreeSwtich，注意版本 比如是 debug 版本

然后下载安装包

wanpipe_6_0_47_1.zip

解压 到 c:\ 然后

cd C:\wanpipe_6_0_47_1\card

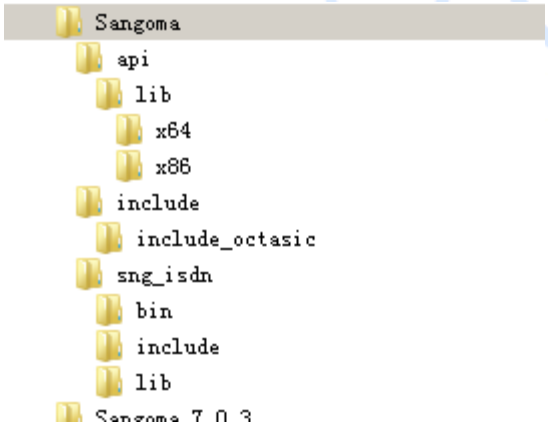
然后

setup.exe install_devel_files

在下载 libsng_isdn-7.6.2-win32.exe

安装。安装之后应该 在 C:\Program Files\Sangoma

目录下有下面这些目录



然后 使用 vs2010 打开 D:\fs\freeswitch-1.2.12\libs\freetdm 目录下的 freetdm.2010.sln

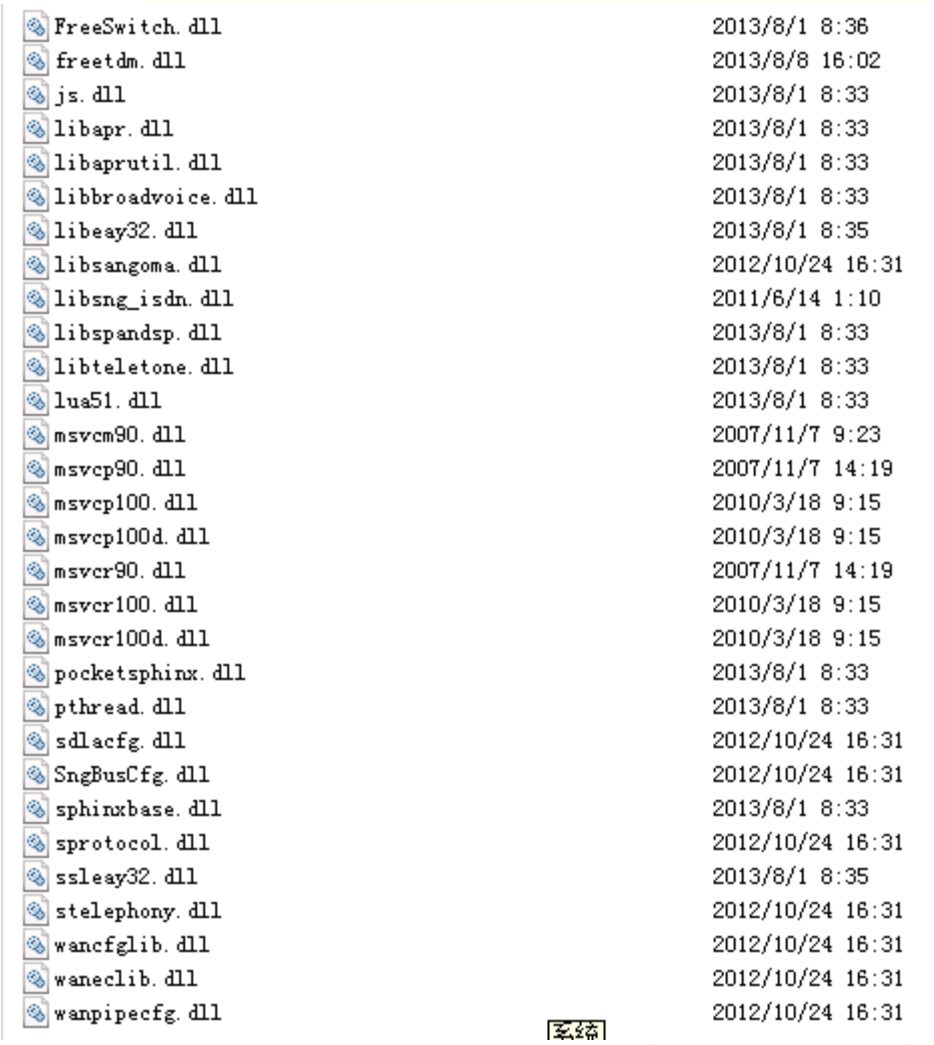
然后选择 解决方案的配置版本，要跟 fs 的版本一致，比如 fs 版本是 debug 版本，那么这里也要选择

debug 版，假如 fs 是 release 版本，那么这里也要选择 release 版本
然后修改
ftmod_wanpipe 工程，把
C:\Program Files\Sangoma\include\include_octasic
加到工程的 附加包含目录下，如下图：

然后按 F7 编译
再 手工编译
ftmod_wanpipe 工程
和 ftmod_sangoma_isdn 工程

编译成功之后产生：
以 release 版本为例：
D:\fs\freeswitch-1.2.12\libs\freetdm\Win32\Release 目录下产生
freetdm.dll
ftmod_wanpipe.dll
ftmod_sangoma_isdn.dll
D:\fs\freeswitch-1.2.12\libs\freetdm\Win32\Release\mod
mod_freetdm.dll
然后拷贝
freetdm.dll 到
D:\fs\freeswitch-1.2.12\Win32\Release
拷贝
ftmod_wanpipe.dll
ftmod_sangoma_isdn.dll

mod_freetdm.dll 到
D:\fs\freeswitch-1.2.12\Win32\Release\mod
然后把 C:\Program Files\Sangoma 安装 目录下的 libsng_isdn.dll 之类文件也拷贝到
D:\fs\freeswitch-1.2.12\Win32\Release
libsng_isdn.dll
再到
C:\wanpipe_6_0_47_1
找出 dll 也放到
D:\fs\freeswitch-1.2.12\Win32\Release 如下图



FreeSwitch.dll	2013/8/1 8:36
freetdm.dll	2013/8/8 16:02
js.dll	2013/8/1 8:33
libapr.dll	2013/8/1 8:33
libaprutil.dll	2013/8/1 8:33
libbroadvoice.dll	2013/8/1 8:33
libeay32.dll	2013/8/1 8:35
libsangoma.dll	2012/10/24 16:31
libsng_isdn.dll	2011/6/14 1:10
libspandsp.dll	2013/8/1 8:33
libteletone.dll	2013/8/1 8:33
lua51.dll	2013/8/1 8:33
msvcm90.dll	2007/11/7 9:23
msvcp90.dll	2007/11/7 14:19
msvcp100.dll	2010/3/18 9:15
msvcp100d.dll	2010/3/18 9:15
msvcr90.dll	2007/11/7 14:19
msvcr100.dll	2010/3/18 9:15
msvcr100d.dll	2010/3/18 9:15
pocketsphinx.dll	2013/8/1 8:33
pthread.dll	2013/8/1 8:33
sdlacfg.dll	2012/10/24 16:31
SngBusCfg.dll	2012/10/24 16:31
sphinxbase.dll	2013/8/1 8:33
sprotocol.dll	2012/10/24 16:31
ssleay32.dll	2013/8/1 8:35
stelephony.dll	2012/10/24 16:31
wancfglib.dll	2012/10/24 16:31
waneclib.dll	2012/10/24 16:31
wanpipecfg.dll	2012/10/24 16:31

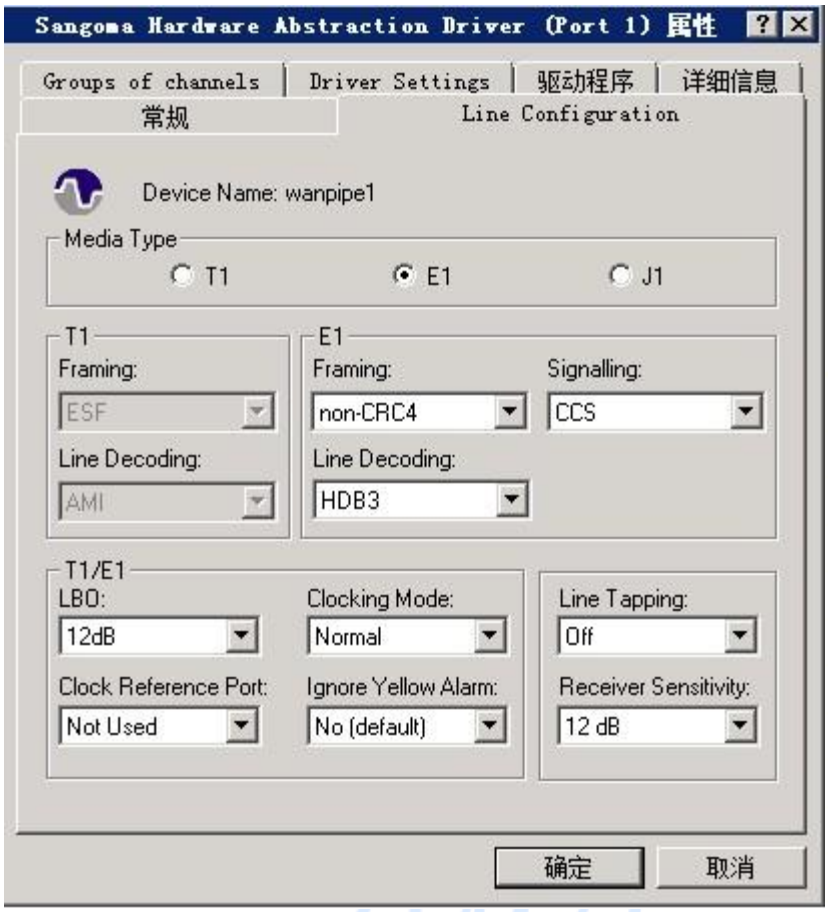
这样就 FreeSwtich 有了 mod_freetdm.dll
这个只是准备（在开发机器上），由于还没有安装驱动和配置卡，FreeSwtich 还是是无法使用 mod_freetdm 的。

217. Windows 下 Sangoma 板卡驱动如何安装和配置 ISDN-PRI 信令？

Sangoma 板卡 Windows 下的安装。参照：
http://wiki.sangoma.com/wanpipe-api-freetdm-windows#wanconfig

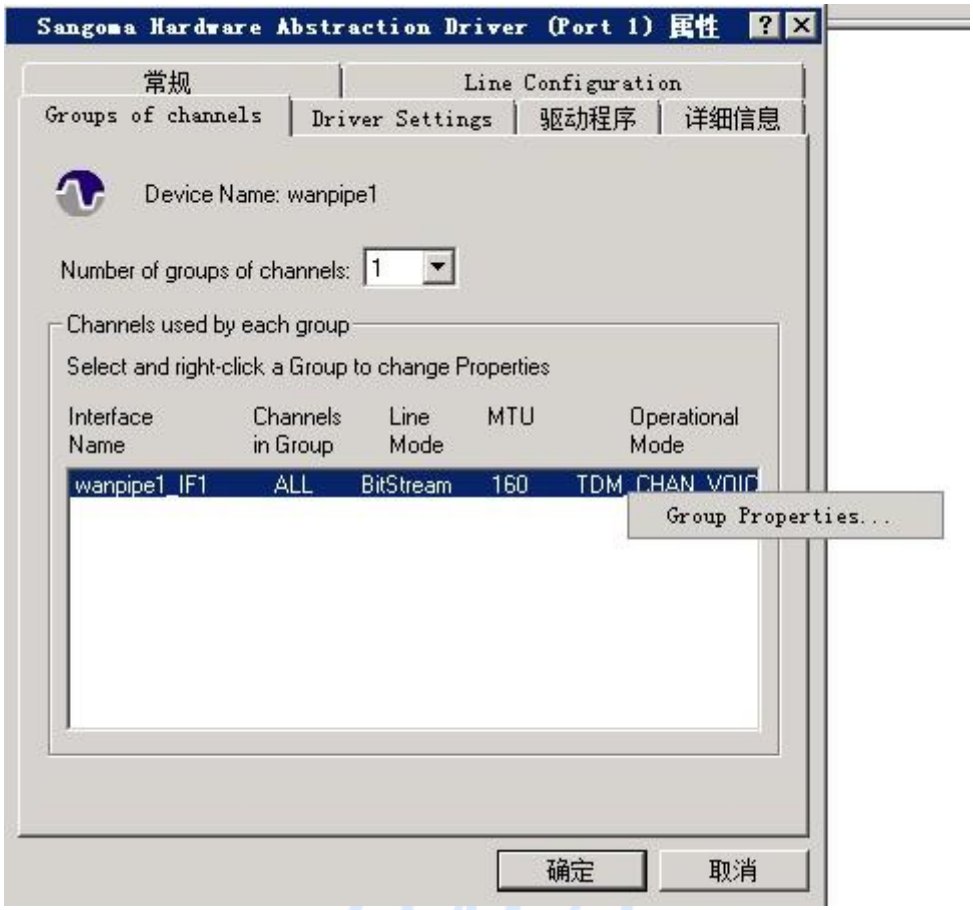
不得不说，Sangoma 网站在中国这边打开巨慢之无比。
以 windows 2008 64 位企业版为例。
先把卡插到机器上。
下载安装包
wanpipe_6_0_47_1.zip
解压 到 c:\，注意官网提供 wanpipe-win-7.0.3.0.zip 的安装，安装之后无法使用。
然后
cd C:\wanpipe_6_0_47_1\card
然后
setup64.exe install_devel_files
再
setup64.exe install
然后重启机器
然后
到设备管理器里面

选择第一个 PORT1，点右键
然后点属性：
然后选择线路配置页面，按照如下配置：

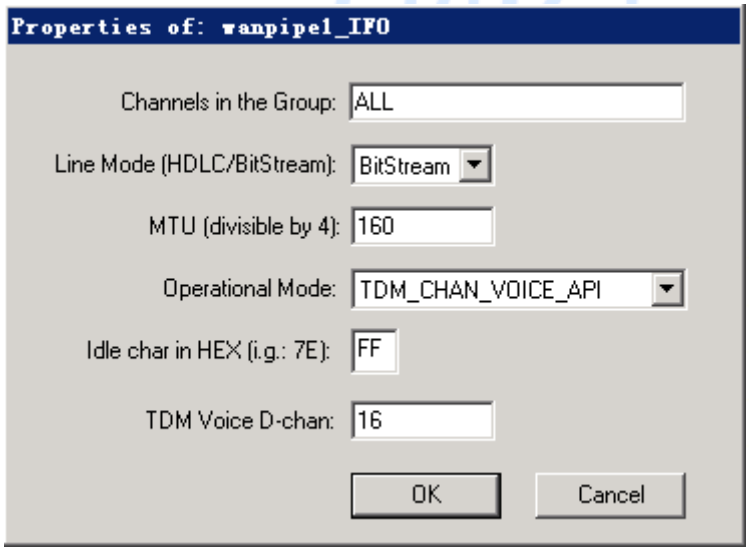


配置 E1，HD3 和 CCS 等必须参数。

其他的主要是根据运营商的线路配置 CRC4，假如线路没有 CRC4，选择 non-CRC4，再选择 Groups of channels 页面 点 右键，然后点 Group properties



然后按照下面配置 properties 页面



然后点 ok，再点确定

出现下面窗口的时候

点是。

然后配置其他 port2 。。。一直到 port8 一个 port 对应一个 E1。

都配置好之后，拷贝上面产生的带 mod_freetdm.dll 的 FreeSwitch 程序到本机器上，

然后配置拨号计划之后（参照 linux 下的安装配置），主要是这句：

<action application="bridge" data="freetdm/wp1/a/$1"/>

然后 修改 conf\autoload_configs\modules.conf.xml

放开

<load module="mod_freetdm"/>

的注释

然后准备配置文件 wanpipe.conf，freetdm.conf 和 freetdm.conf.xml (这些配置文件来自 linux 版的配置文件）

在 FreeSwitch 的 conf\ 目录下 wanpipe.conf 和 freetdm.conf 文件

wanpipe.conf 内容如下

```
[defaults]
; User space interval at which data will be delivered
codec_ms => 20

; wink and flash interval
wink_ms => 150
flash_ms => 750

; size of the driver queue of elements of MTU size
; typical case is 10 elements of 80 bytes each (10ms of ulaw/alaw)
; don't mess with this if you don't know what you're doing
; txqueue_size => 10
; rxqueue_size => 10
```

freetdm.conf 内容如下

```
[span wanpipe wp1]
trunk_type => e1
group=1
b-channel => 1:1-15
b-channel => 1:17-31
d-channel => 1:16
[span wanpipe wp2]
trunk_type => e1
group=1
b-channel => 2:1-15
```

```
b-channel => 2:17-31
d-channel => 2:16
[span wanpipe wp3]
trunk_type => e1
group=1
b-channel => 3:1-15
b-channel => 3:17-31
d-channel => 3:16
[span wanpipe wp4]
trunk_type => e1
group=1
b-channel => 4:1-15
b-channel => 4:17-31
d-channel => 4:16
[span wanpipe wp5]
trunk_type => e1
group=1
b-channel => 5:1-15
b-channel => 5:17-31
d-channel => 5:16
[span wanpipe wp6]
trunk_type => e1
group=1
b-channel => 6:1-15
b-channel => 6:17-31
d-channel => 6:16
[span wanpipe wp7]
trunk_type => e1
group=1
b-channel => 7:1-15
b-channel => 7:17-31
d-channel => 7:16
[span wanpipe wp8]
trunk_type => e1
group=1
b-channel => 8:1-15
b-channel => 8:17-31
d-channel => 8:16
```

FreeSwitch 的 conf\autoload_configs 目录下 freetdm.conf.xml 文件。

freetdm.conf.xml 文件 内容如下：

```
<configuration name="freetdm.conf" description="Freetdm Configuration">
  <settings>
    <param name="debug" value="0"/>
    <param name="sip_headers" value="false"/>
```

```
        <!--<param name="hold-music" value="$${moh_uri}"/>-->
        <!--<param name="enable-analog-option" value="call-swap"/>-->
        <!--<param name="enable-analog-option" value="3-way"/>-->
    </settings>
     <config_profiles>
          <profile name="my_pri_te_1">
                                <param name="switchtype" value="euroisdn" />
                                <param name="interface" value="pri_cpe"/>
                               <param name="facility" value="no" />
                               <param name="outbound-called-ton" value="national"/>
                <param name="outbound-called-npi" value="isdn"/>
                <param name="outbound-calling-ton" value="national"/>
                <param name="outbound-calling-npi" value="isdn"/>
                <param name="outbound-rdnis-ton" value="national"/>
                <param name="outbound-rdnis-npi" value="isdn"/>
                <param name="outbound-bearer_cap" value="speech"/>
                <param name="outbound-bearer_layer1" value="alaw"/>
        </profile>
</config_profiles>
            <sangoma_pri_spans>
                    <span name="wp1" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
                    <span name="wp2" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
                    <span name="wp3" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
                    <span name="wp4" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
                    <span name="wp5" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
                    <span name="wp6" cfgprofile="my_pri_te_1">
                            <param name="dialplan" value="XML"/>
                            <param name="context" value="default"/>
                    </span>
```

```
                <span name="wp7" cfgprofile="my_pri_te_1">
                        <param name="dialplan" value="XML"/>
                        <param name="context" value="default"/>
                </span>
                <span name="wp8" cfgprofile="my_pri_te_1">
                        <param name="dialplan" value="XML"/>
                        <param name="context" value="default"/>
                </span>
        </sangoma_pri_spans>
        <analog_spans>
        </analog_spans>
</configuration>
```

注意：Linux 版本下的 wanpipe1 到 wanpipe8.conf 这些文件不需要。

然后启动 FreeSwitch,接上 E1 线路。

然后再使用命令看 E1 的状态

```
freeswitch@WIN-6DS8EHJLLUC> ftdm sangoma_isdn show_spans
span:wp1 physical:OK signalling:UP
span:wp2 physical:OK signalling:UP
span:wp3 physical:OK signalling:UP
span:wp4 physical:OK signalling:UP
span:wp5 physical:OK signalling:UP
span:wp6 physical:OK signalling:UP
span:wp7 physical:OK signalling:UP
span:wp8 physical:OK signalling:UP
Command executed OK
```

可以使用软电话注册上来呼叫测试。

218. Sangoma 板卡常用命令有哪些？

Sangoma 板卡 Linux 版本（windows 版本下有些命令无法使用，比如 wanrouter 命令）下常用的命令有：

wanrouter 命令和 wanpipemon 命令

停止驱动 #wanrouter stop

重启驱动 wanrouter restart

查看物理层连接情况：

#wanrouter status

```
Devices currently active:
        wanpipe1
Wanpipe Config:
Device name | Protocol Map | Adapter    | IRQ | Slot/IO | If's | CLK | Baud rate |
wanpipe1 | N/A| A101/1D/2/2D/4/4D/8/8D/16/16D| 17    | 4    | 1 | N/A | 0        |
Wanrouter Status:
Device name | Protocol | Station | Status              |
wanpipe1       | AFT TE1    | N/A        | Connected        |
```

```
[root@sipserver wanpipe]# wanpipemon -i w1g1 -c Ta
***** w1g1: E1 Rx Alarms (Framer) *****
ALOS:     OFF         | LOS:    OFF
RED:      OFF         | AIS:    OFF
LOF:      OFF         | RAI:    OFF
***** w1g1: E1 Rx Alarms (LIU) *****
Short Circuit:    OFF
Open Circuit:     OFF
Loss of Signal: OFF
***** w1g1: E1 Tx Alarms *****
AIS:      OFF         | YEL:    OFF
***** w1g1: E1 Performance Monitoring Counters *****
Line Code Violation         : 575
Far End Block Errors        : 0
CRC4 Errors                 : 0
FAS Errors                  : 0
Sync Errors                 : 0
Rx Level              : > -2.5db
```

freeswitch 里面的操作命令：

列出每个 E1 状态

```
freeswitch@sipserver> ftdm list
+OK
span: 1 (wp1)
type: Sangoma (ISDN)
physical_status: ok
signaling_status: UP
chan_count: 31
dialplan: XML
context: default
dial_regex:
fail_dial_regex:
hold_music:
analog_options: none
```

查看信令状态：

```
freeswitch@sipserver> ftdm sangoma_isdn show_spans

span:wp1 physical:OK signalling:UP
Command executed OK
```

可以看 isdn 上的抓包信令状态：

```
freeswitch@sipserver>ftdm sangoma_isdn trace q931    wp1
```

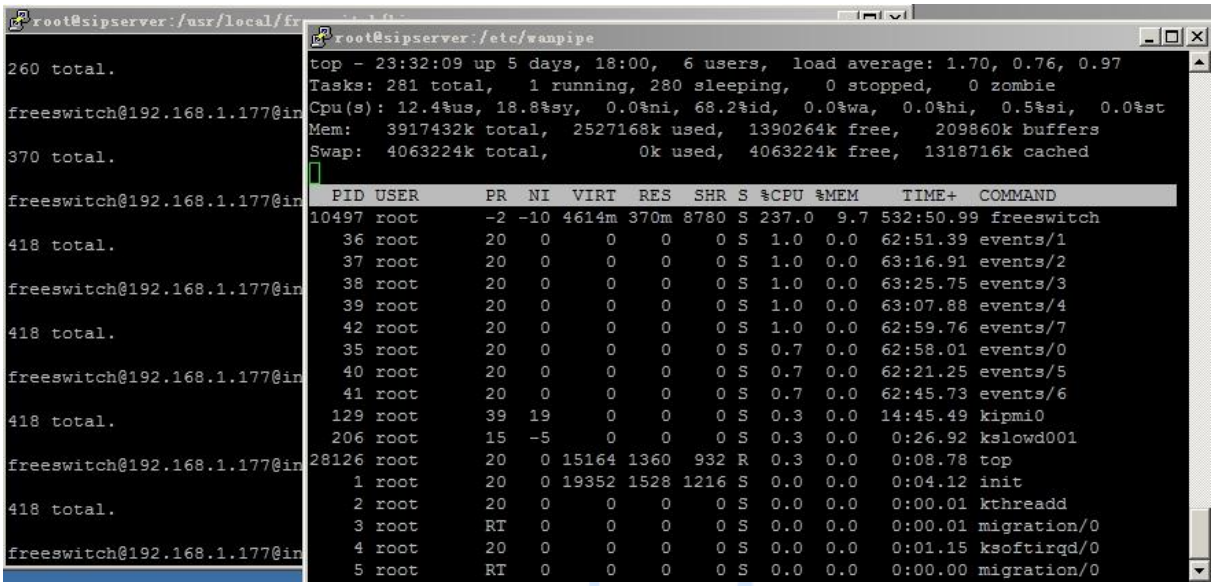
219. Sangoma 板卡压力测试性能如何？

Sangoma 板卡压力测试 8E1 的卡 240 路并发：

测试场景 1：

210 路 sip 拨入使用 alaw 编码，210 路 TDM 转出通话，总共 420 路，cpu 占用 30%。

测试环境 Linux Centos 6.4 版本 FreeSwitch 1.2.12 CPU E5405 *2， 内存 4G，如下图：



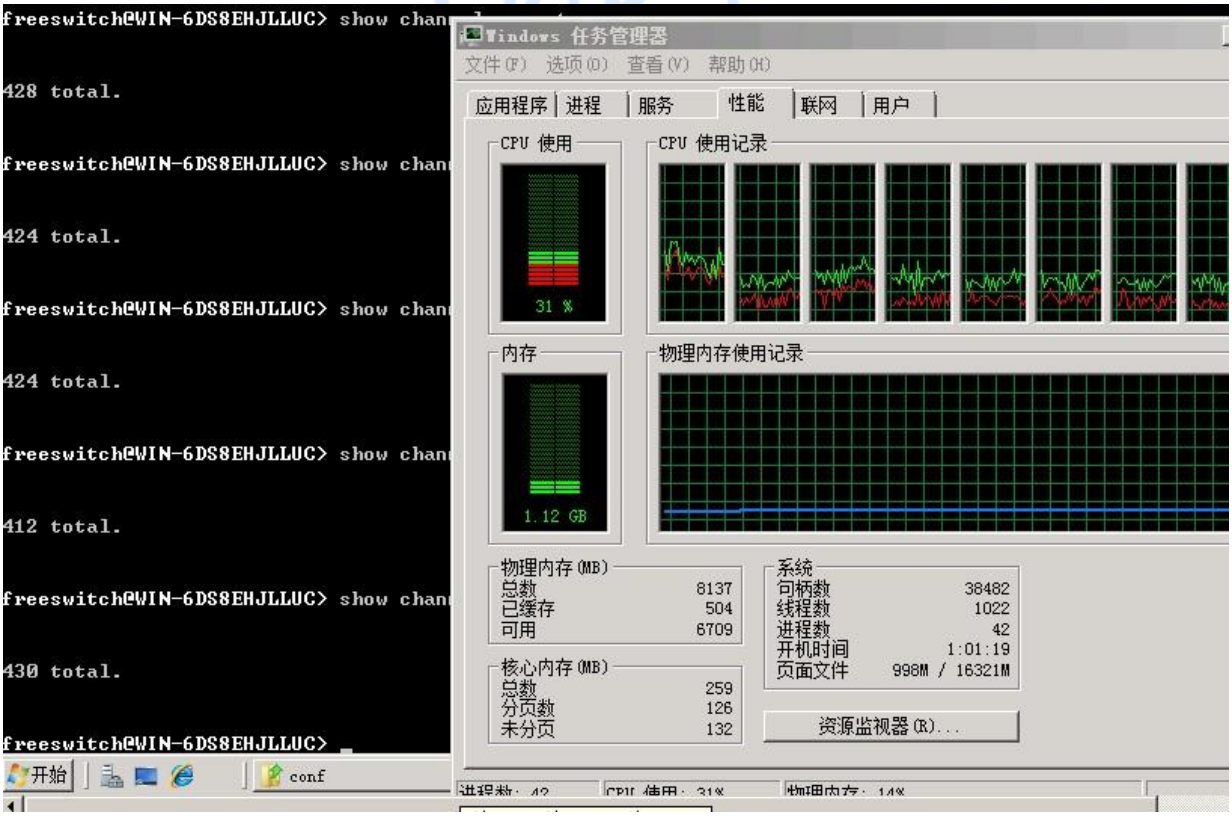
测试场景 2：

215 路 sip 拨入使用 alaw 编码，215 路 TDM 转出通话，总共 430 路左右，cpu 占用 31%

测试环境：

Windows 2008 64 位 版本 FreeSwitch 1.2.12 32 位编译 CPU E3-1230V2， 内存 8G

CPU 使用如下：



测试运行接近 10 个小时，通话 87712 次，然后停止测试，发现 FreeSwitch 存在句柄泄漏，如下图，句柄占用达到了 177,999 个（难道拨入转出的呼叫由于有两个 channels，因此每次呼叫泄漏 2 个？）。

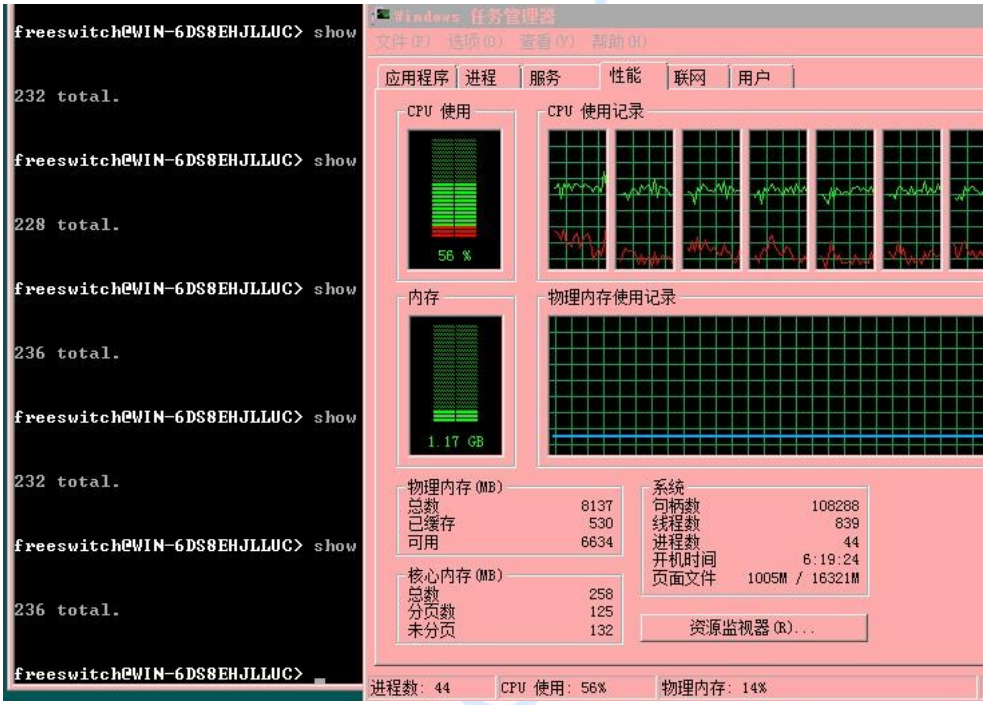
测试场景 3：

120 路 sip 拨入使用 iLBC 编码，120 路 TDM 转出通话，总共 240 路左右，cpu 占用 55%

测试环境：

Windows 2008 64 位 版本 FreeSwitch 1.2.12 32 位编译 CPU E3-1230V2， 内存 8G

CPU 使用如下：



对比上面测试场景 2 发现语音转码的 CPU 开销还是很大的：线路少了一半，cpu 开销几乎多了一倍，换句话说转码的 cpu 开销是没有转码的 cpu 开销的差不多 3-4 倍。

175 sip 拨入使用 iLBC 编码，175 路 TDM 转出通话，总共 350 路左右，cpu 占用 75%左右,偶尔有错误输出：

freeswitch@WIN-6DS8EHJLLUC> 2013-08-26 16:47:18.861799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/8:2/8000008 device 8:2!

2013-08-26 16:53:11.711799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/3:10/8000003 device 3:10!

2013-08-26 17:17:17.411799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/8:8/8000008 device 8:8!

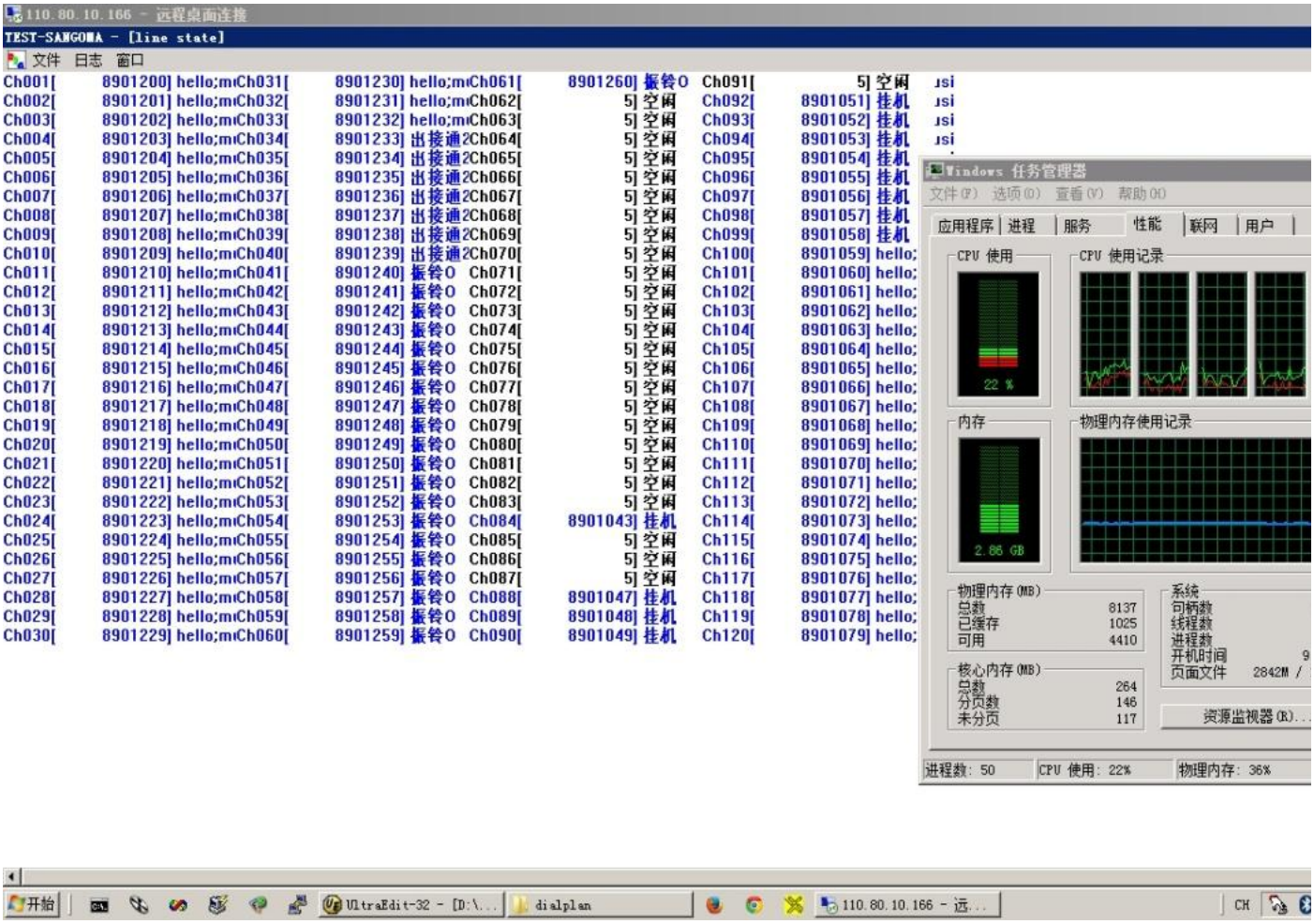
2013-08-26 17:17:35.641799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/7:7/8000007 device 7:7!

2013-08-26 17:34:01.681799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/6:7/8000006 device 6:7!

2013-08-26 17:42:16.181799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/6:19/8000006 device 6:19!

2013-08-26 17:48:29.543799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/3:4/8000003 device 3:4!
2013-08-26 18:31:05.521799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/7:9/8000007 device 7:9!
2013-08-26 18:37:14.494799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/6:8/8000006 device 6:8!
2013-08-26 18:53:48.461799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/7:4/8000007 device 7:4!
2013-08-26 19:15:14.185799 [ERR] mod_freetdm.c:824 clearing IO in channel FreeTDM/2:2/8000002 device 2:2!

测试场景 4：通过 ESL 发起 TDM 的呼叫通话，然后系统进入 IVR 播音，播放 alaw 格式的语音，取按键，并发在 210 路左右，cpu 占用 22%左右。
测试环境：
Windows 2008 64 位 版本 FreeSwitch 1.2.12 32 位编译 CPU E3-1230V2， 内存 8G



220. Sangoma 板卡拨出如何支持指定线路呼叫？

Sangoma 板卡 外拨的时候，可以让板卡自己选择空闲通道外拨，比如：
<action application="bridge" data="freetdm/wp1/a/13606060253"/>
是然板卡自己选择第 1E1 的端口里面从 1-30 路自己选择通道外拨，
这样会存在问题:通道的使用不是轮换模式，而是每次都从第 1 路开始，只有第 1 路忙才会选择第 2 路。另外假如第 1E1 全部都是忙，切换不到第 2E1.另外。
<action application="bridge" data="freetdm/wp1/A/13606060253"/> 表示
先选 30 路，30 路忙再考虑 29 路。。。。

实际上，是可以自己指定某路外拨的：
<action application="bridge" data="freetdm/wp1/1/13606060253"/>
表示指定使用第 1E1 的第 1 路外拨。
<action application="bridge" data="freetdm/wp2/30/13606060253"/>
表示指定使用第 2E1 的第 30 路外拨。

221. ESL 如何支持 Sangoma 板卡拨入和拨出？

ESL 上支持 Sangoma 卡的核心是自己维护一个线路的状态表。
比如 0-239 对应 1-8E1 的每个电路：
ESL 拨入：

```
static void mycallback(esl_socket_t server_sock, esl_socket_t client_sock, struct sockaddr_in *addr)
{
。。。
if(_strnicmp(channel_name,"FreeTDM",7)==0)
 {
        //FreeTDM/8:11/91001
        char *p;
        int span,slot;
        p=strstr(channel_name,":");
        if(p==NULL) goto DROPCALL;
        strcpy(tmp,channel_name+8);
        strtok(tmp,":");
        span=atoi(tmp);
        strcpy(tmp,p+1);
        strtok(tmp,"/");
        slot=atoi(tmp);
 index=(span-1)*30+(slot-1);//index start from 0 ,slot start from 1, span start from 1
        linestate[index]=1;//     标识线路状态为忙
 disp_msg("index=%03d,calldir=%d,uuid=%s,caller=%s,called=%s,span=%d,slot=%d,
                channel_name=%s",index,dir,uuid,caller,called,span,slot,channel_name);
 }
```

ESL 拨出：
自己指定某路外拨需要自己进行外拨路由的选择和控制，方法如下：
先选择拨出的 E1 的空闲线路，算法如下：

```
for(i=0;i<240;i++;) if (linestate[i]==0) break;
if(i>=240) return -1;//无空闲线路，返回错误
index=i;
 //说明：index 从 0 开始
 int span=(index-freetdm_start_index)/30+1;
 int slot=(index-freetdm_start_index)%30+1;                    sprintf(cmd_tmp,"bgapi originate
{origination_uuid=%s,originate_timeout=%d,origination_caller_id_number=%s,origination_caller_id_name
```

```
=%s}freetdm/wp%d/%d/%s %s\n\n",uuid,timer,caller,caller,span,slot,called,system_caller);
    res=esl_send_recv_timed(&handle,cmd_tmp,1000);
```

222. Sangoma 板卡如何设置拨出主叫被叫的号码属性？

Sangoma 板卡可以设置呼叫出去的被叫号码的属性，具体方法如下：
在/usr/local/freeswitch/conf/autoload_configs/freetdm.conf.xml 文件里面
里面设置如下参数：

```
<config_profiles>
        <profile name="my_pri_te_1">
            <param name="switchtype" value="euroisdn" />
            <param name="interface" value="pri_cpe"/>
            <param name="facility" value="no" />
            <param name="outbound-called-ton" value="national"/>
             <param name="outbound-called-npi" value="isdn"/>
             <param name="outbound-calling-ton" value="national"/>
             <param name="outbound-calling-npi" value="isdn"/>
             <param name="outbound-rdnis-ton" value="national"/>
             <param name="outbound-rdnis-npi" value="isdn"/>
             <param name="outbound-bearer_cap" value="speech"/>
             <param name="outbound-bearer_layer1" value="alaw"/>
         </profile>
    </config_profiles>
```

本人测试 ISDN PRI 根据文档在 freetdm.conf.xml 里面：设置下面参数：

```
    <param name="sip_headers" value="true"/>
```

然后 在拨号计划里面 里面设置：

```
<action application="set" data="sip_h_X-FreeTDM-DNIS-TON=2"/>
<action application="set" data="sip_h_X-FreeTDM-DNIS-Plan=1"/>
<action application="set" data="sip_h_X-FreeTDM-DNIS-NADI=1"/>
<action application="set" data="sip_h_X-FreeTDM-ANI-TON=2"/>
<action application="set" data="sip_h_X-FreeTDM-ANI-Plan=1"/>
<action application="set" data="sip_h_X-FreeTDM-ANI-NADI=1"/>
```

是不行的。

223. Sangoma 板卡有哪些需要注意的？

Sangoma 卡驱动启动之后，ISDN PRI 信令不会启动，只有 FreeSwitch 程序启动正常之后，ISDN PRI 信令才会 UP，对端看 ISDN PRI 才是启动的状态。假如没有启动 FreeSwitch 程序，对方看的就是 DOWN 的状态。

224. Sangoma 板卡带 DSP 跟不带 DSP 有啥区别？

Sangoma 卡有两种，一种带 DSP，一种没有带 DSP，带 DSP 价格几乎贵了一倍。除了回声消除提高语音质量之外，他们的还有啥区别呢？总代理的老朱也说不出啥区别。

我实际使用过程中有发现一个区别，在拨出取按键的时候有区别，假如拨出到用户手机上，通话比较久之后，没有带 DSP 的卡取按键就可能不成功。

第十三章 IMS 部分

225. FreeSwitch 部署为啥需要支持 IMS？

最近几年运营商开始推广 IMS 系统，直接拉一个专线到宽带某个公司，提供一个 IP，再提供一批 SIP 帐号（电话号码）和密码，就可以使用。

对公司运营而言，只要一个普通的 pc server 就可以构建一套系统。省去了板卡采购的成本。这样一个普通的 1 万多元 PC Server 可以支持很容易 1000 线以上,甚至 2000 线也不在话下，相当于每线 5-10 元 RMB。

公司有了这些帐号，可以在他们的专线的 IP 上使用，甚至也可以在阿里云的公网使用这些帐号进行使用。

226. FreeSwitch 如何配置才能使用 IMS？

在 conf\sip_profiles\external 目录下创建 gw1.xml 文件

网关配置内容如下：

```
<gateway name="gw1">
      <param name="realm" value="ims.fj.chinamobile.com"/>
      <param name="from-domain" value="ims.fj.chinamobile.com"/>
      <param name="proxy" value="ims.fj.chinamobile.com"/>
      <param name="outbound-proxy" value="211.143.147.XX"/>
      <param name="register-proxy" value="211.143.147.XX"/>
      <param name="username" value="8659238820XX@ims.fj.chinamobile.com"/>
      <param name="from-user" value="+8659238820XX"/>
      <param name="password" value="XXXXXXXXX"/>
      <param name="register" value="true" />
      <param name="extension" value="+8659238820XX"/>
</gateway>
```

配置之后就可以通过 gw1 来使用 ims。

假如多个号码就配置多个 gw,比如 gw2，gw3....然后轮流使用这些 gw。

227. FreeSwitch 如何配置才支持 IMS 拨入系统 IVR？

IMS 拨入的时候拨入主叫：+86592xxxxxxx， 因为是+开头的号码要修改 public.xml 的拨号计划，否则不行。 在 public.xml 里面把+吃掉。

拨号计划里面修改：public.xml

```
<extension name="public_extensions">
    <condition field="destination_number" expression="^[+|#|A|T](\d+)$">
  <action application="transfer" data="$1 XML default"/>
    </condition>
```

```
    </extension>
<extension name="public_extensions">
      <condition field="destination_number" expression="^([0-9]\d+)$">
    <action application="transfer" data="$1 XML default"/>
      </condition>
    </extension>
```

假设拨入到 ESL 上，default.xml 拨号计划部分如下：

```
    <extension name="Local_Extension86">
          <condition field="destination_number" expression="(^86\d+)$">
                        <action application="export" data="dialed_extension=$1"/>
                        <action application="set" data="call_timeout=30"/>
                          <action application="set" data="record_sample_rate=8000"/>
                          <action application="export" data="RECORD_STEREO=false"/>
                        <action application="set" data="hangup_after_bridge=true"/>
                        <action application="set" data="continue_on_fail=true"/>
                        <action application="start_dtmf" />
                        <action application="socket" data="127.0.0.1:8084 async full"/>
          </condition>
        </extension>
```

注意：<action application="start_dtmf" />

表示启动 DTMF 按键检测，没有这个检测不到按键。

其他说明：

拨入容量限制：每个号码实际测试可以超过 10 线并发,最大到 16 线并发。

228. FreeSwitch 如何配置才支持软电话通过 IMS 转出系统？

IMS 的拨出最大每个号码允许两线并发，测试发现 IMS 同样的一个号码拨出，

假如很快发起多个呼叫会收到 487 fail to modify user state

因此需要多个号码轮流循环使用。假设 有 abcde 5 个号码，需要这样使用：abcde abcde,

假如是 aabbccddee 可能会呼叫失败。

拨号计划配置如下：

```
    <extension name="Local_Extension21">
          <condition field="destination_number" expression="^([0-9]\d+)$">
                        <action application="export" data="dialed_extension=$1"/>
                        <action application="set" data="call_timeout=30"/>
                          <action application="set" data="record_sample_rate=8000"/>
                          <action application="export" data="RECORD_STEREO=false"/>
                        <action application="set" data="hangup_after_bridge=true"/>
                        <action application="set" data="continue_on_fail=true"/>
                        <action application="answer" />
                        <action application="sleep" data="1000"/>
                        <action application="start_dtmf" />
```

```
                <action application="playback" data="d:/data/system/music3.alaw"/>
                    <action application="bridge" data="sofia/gateway/gw1/$1"/>
                </condition>
    </extension>
```

注意：

拨出过程中假如软电话有按键，IMS 马上结束呼叫。

软电话拨入转出，拨号计划要先应答，这样软电话才能听见用户的

彩铃之类的振铃 的声音。

229. IMS 系统支持哪些语音编码器？

IMS 测试支持的编码器有：alaw，ulaw，G729，

IMS 测试不支持的编码器：iLBC，gsm，speex,

由于 IMS 的 DTMF 按键是 inband 代码方式传输，因此

假如是公网使用有两种考虑：

假如不是 IVR 系统，用户不需要按键选择，那么跟 IMS 对接，使用 G729。

假如是 IVR 系统，需要取用户的按键，那么跟 IMS 对接使用 alaw 或者 ulaw。

假如是专线网络使用，那么跟 IMS 对接使用 alaw 或者 ulaw。

另外由于 FreeSwitch 默认不支持 G729 的转码，因此要么不使用 G729，要么使用支持转码的 FreeSwitch 版本，假如有需要支持 G729 转码的 FreeSwitch 版本可联系本人。本人提供基于 IPP 转码的 G729 动态库。

230. FreeSwitch 如何配置才通过 IMS 系统支持收发传真？

收传真的拨号计划如下：

```
<extension name="fax_receive">
        <condition field="destination_number" expression="^([0-9]\d+)$">
<action application="answer" />
            <action application="set" data="fax_enable_t38=true"/>
            <action application="set" data="fax_enable_t38_request=true"/>
<action application="playback" data="silence_stream://2000"/>
<action application="rxfax" data="c:/test_${strftime(%Y-%m-%d-%H-%M-%S)}.tif"/>
<action application="hangup"/>
        </condition>
    </extension>
```

发传真的测试命令如下：

originate sofia/gateway/gw1/059259988XX &txfax(c:/rxfax0.tif)

实际测试支持多页传真的收发。

第十四章 WebRTC（WEBPhone）部分

231. WebRTC 是啥东东？

我的个人理解 WebRTC 就是让 web 浏览器也可以进行 voip，也相当于把浏览器作为 voip 的软电话。我给它起一个通俗的名称：WEB Phone
大家知道没有一台个人电脑上不会不带浏览器。而之前视频通话都是需要客户端软件支持的。假如浏览器能直接视频通话，这个就让用户的 VOIP 视频从天上变成了人间。
一个 qq 的群主说：“WebRTC 是 VOIP 最后的疯狂。”
基本上 WebRTC 是通过页面上的 js 一些 api 控制支持电话的发起呼叫，应答和挂机等信令控制，这个通常是 websocket 的方式走 tcp 控制信令。
再通过浏览器里面内置的语音和视频编解码器来实现媒体。
这个是经典的 rtp 走 udp 方式。
说白了 WebRTC 就是使用浏览器作为 VOIP 软电话。

232. WebRTC 有哪些开源客户端？

由于 WebRTC 采用了 SDP 方式媒体协商。
带来了经典的 VOIP 的臭问题：NAT 穿透。
而 WebRTC 的设计初衷是 P2P 模式。
再加上 P2P 模式下的 NAT 穿透是最困难的，因此又引入了使用 ICE 来穿透。
结果问题越来越大。
因此有人说 WebRTC 是好点子，但是被一群人使用愚蠢的办法来实现。

跑题了。回到 FreeSwitch。
FreeSwitch 1.4.X 开始支持 WebRTC，也就是说 FreeSwitch 1.4.X 可以作为 WebRTC 服务端。我测试是使用 1.4.6 版本测试。

但是 WebRTC 客户端需要用户自己使用 js api 基于 HTML5 编写，
这个方面的有关介绍比较少，
有一本书供参考： High Performance Browser Networking 高性能浏览器网络（影印版）。
里面有专门一个章节讲 websocket 还有一个专门的章节讲 WebRTC。
该书某东上有销售。

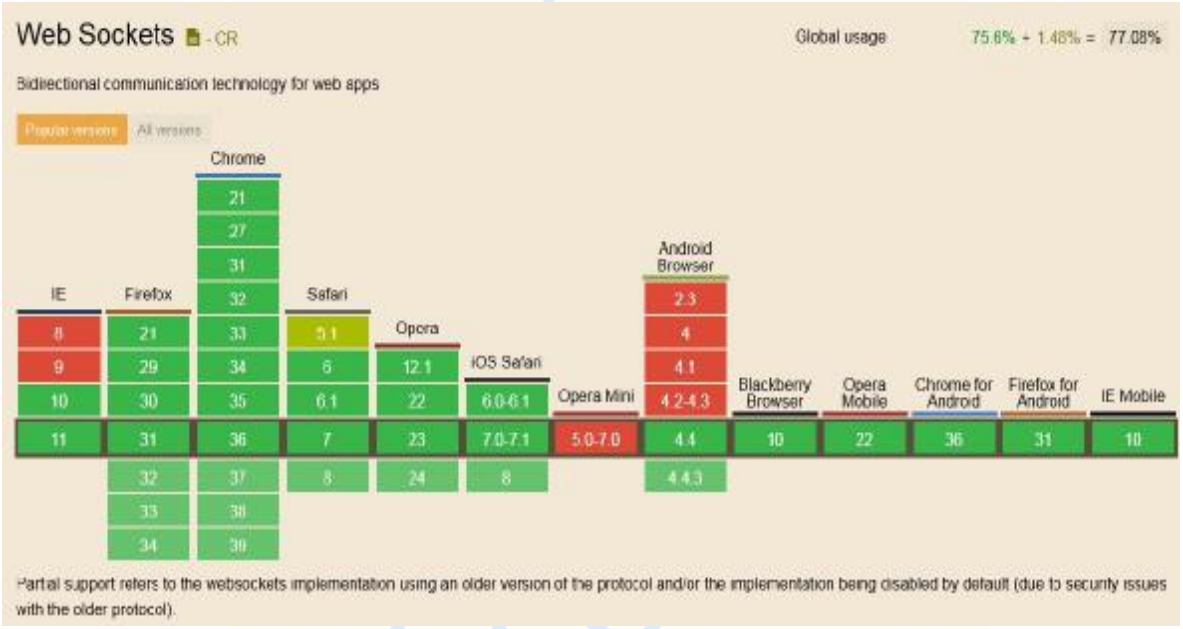
WebRTC 实际上很多开发是客户端的开发工作量，只不过可以利用已经有了开源的，客户端开发的工作量就少了很多。否则像我这种做后台开放的人来做 WEB 的 JS API 开发简直就是天方夜谭。
假如伙计们对 JS 开发不熟悉，要做这个 要么找一个做 web 的伙计配合，要么自己赶快恶补一下 JS 的开发。

感谢万能的 Google 万能的开源，你要的东东除了女人，毒品和军火之外啥都能提供。
目前本人测试通过的能用的 WebRTC 客户端开源的已经有两个：
sipML5 网站 https://code.google.com/p/sipml5/
和
JSSip 网站 http://www.jssip.net/

关于源码：
sipML5 上有源码可以直接下载。sipML5 比较庞大，测试 sipML5 有些问题。
JSSip 比较精干，不过它的源码需要花点办法折腾整合，修改一些 bug 就基本可以使用。
测试基本上没有啥大问题。而且包也比较小。需要的同学可以联系我。以下的测试均以 JSSIP 为蓝本进行。

233. 哪些浏览器支持 WebRTC?

目前情况下要支持 WebRTC，首先需要支持 WebSockt。下面这个图是从 JsSip 网站上搞到的。



绿色代表支持 WebSocket。
看起来这个星球上的主流浏览器好像基本都是支持的。
不过别忘记还需要媒体的支持，比如安卓 4.4.2 自带的浏览器，使用 webrtc 可以注册，但是呼叫不行。视频通话更加无从谈起。
我测试下来目前就火狐和 chrome 支持比较好。
本人测试的版本：
Windows 平台：
chrome 版本 36
火狐版本 31
安卓平台：
chrome 版本 36
火狐版本 31
都是可以使用的。

假如你要使用安卓测试，请注意安卓的版本。
安卓的版本假如是 2.3 的版本，chrome 不能安装，火狐可以安装但是使用也有问题。
我建议安卓使用 4.4.X 版本安装 chrome 或者火狐测试。
只要这些版本或者之上的版本，打开网站就可以视频。不需要 7788 的插件和浏览器设置。视频通话前，浏览器会弹出对话框，让用户选择是否共享麦克风和视频头，要选择允许才能通话，假如选择拒绝就不能通话，如下图：
Chrome 的共享：

假如是拨入到 WEB Phone，Chrome 的共享只要设置一次就可以，将来不需要设置。
这个对视频呼叫中心的业务开发比较有利，否则每次都要共享，操作难免太麻烦。

火狐 的共享界面：

火狐出现这个界面的时候，不能切换到其他页面去，假如这个时候你切换到其他界面 就麻烦了，只能重新刷新页面，重新登录和呼叫。否则不能使用。
这个使用的过程中要注意。

而对于桌面版本市场占有率最高的 IE，目前貌似还是不支持 WebRTC。
牛逼的微软。呵呵。
实际测试，使用 Win8.1 自带的 IE11 打开上面的页面可以注册登录，可以给其他分机发消息，但是无法发起音频或者视频通话。这说明 IE11 是支持 WebSocket 的，但是不支持 WebRTC。

这个不能不说是 WebRTC 的一个缺憾。导致了 WebRTC 的用户使用场景会有一些限制。设计业务的时候要注意，这个后面会提到。

234. FreeSwitch 如何配置才能支持 WebRTC?

FreeSwitch 支持 WebRTC 很简单，只要做几个事情:

a) 配置 conf\sip_profiles 目录下的 internal.xml

```
<!-- uncomment for sip over websocket support -->
<param name="ws-binding"  value=":5066"/>
```

参数，这个 5066 端口就是浏览器的 websocket 要连接的端口，这个参数默认是没有放开。

b) 配置需要支持的音频和视频编码器，比如需要支持视频

修改 vars.xml 文件里面 配置：

```
<X-PRE-PROCESS cmd="set" data="global_codec_prefs=OPUS,G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM"/>
```

改为：

```
<X-PRE-PROCESS cmd="set" data="global_codec_prefs=OPUS,iLBC,PCMA,GSM,H264,VP8"/>
```

<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=OPUS,iLBC,PCMA,GSM,H264,VP8"/>
假如要视频 VP8 目前是必须的，OPUS 是效果比较好的音频编码器。

235. WebRTC 该如何使用哪些音频和视频编码？

我的个人理解 WebRTC 就是让 web 浏览器也可以进行 voip，也相当于把浏览器作为 voip 的软电话。
通过抓包和日志发现：
Chrome 浏览器的 SDP 如下，超级多的垃圾信息，同学们慢慢看找出有用的：
v=0
o=- 2636115687421352815 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS tiPYTIjN4dBCPJPy0WUo6hoV4cOIuubqww15
m=audio 50032 RTP/SAVPF 111 103 104 0 8 106 105 13 126
c=IN IP4 211.162.33.95
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=rtcp:50032 IN IP4 211.162.33.95
a=candidate:211156821 1 udp 2122194687 192.168.1.5 50032 typ host generation 0
a=candidate:211156821 2 udp 2122194687 192.168.1.5 50032 typ host generation 0
a=candidate:1108738981 1 tcp 1518214911 192.168.1.5 0 typ host generation 0
a=candidate:1108738981 2 tcp 1518214911 192.168.1.5 0 typ host generation 0
a=candidate:2781507712 1 udp 1685987071 211.162.33.95 50032 typ srflx raddr 192.168.1.5 rport 50032 generation 0
a=candidate:2781507712 2 udp 1685987071 211.162.33.95 50032 typ srflx raddr 192.168.1.5 rport 50032 generation 0
a=ice-ufrag:fChVLLHDOF2MRiVr
a=ice-pwd:+1rKCJjEUfSRINYz15Flsi7I
a=ice-options:google-ice
a=fingerprint:sha-256
B4:5F:4D:B3:4A:0C:89:11:EB:48:9A:8D:93:94:35:A7:90:E1:B4:6C:E1:85:7E:04:7D:D5:EA:0F:1E:C3:AC:
a=setup:actpass
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level

a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=rtcp-mux
a=maxptime:60
a=ssrc:3828300439 cname:UYec8Hvix40P60XM
a=ssrc:3828300439
msid:tiPYTIjN4dBCPJPy0WUo6hoV4cOIuubqww15 95126406-9798-456d-9338-712cfc17492c
a=ssrc:3828300439 mslabel:tiPYTIjN4dBCPJPy0WUo6hoV4cOIuubqww15
a=ssrc:3828300439 label:95126406-9798-456d-9338-712cfc17492c
m=video 50032 RTP/SAVPF 100 116 117
c=IN IP4 211.162.33.95
a=rtpmap:100 VP8/90000
a=rtpmap:116 red/90000
a=rtpmap:117 ulpfec/90000
a=rtcp:50032 IN IP4 211.162.33.95
a=candidate:211156821 1 udp 2122194687 192.168.1.5 50032 typ host generation 0
a=candidate:211156821 2 udp 2122194687 192.168.1.5 50032 typ host generation 0
a=candidate:1108738981 1 tcp 1518214911 192.168.1.5 0 typ host generation 0
a=candidate:1108738981 2 tcp 1518214911 192.168.1.5 0 typ host generation 0
a=candidate:2781507712 1 udp 1685987071 211.162.33.95 50032 typ srflx raddr 192.168.1.5 rport 50032 generation 0
a=candidate:2781507712 2 udp 1685987071 211.162.33.95 50032 typ srflx raddr 192.168.1.5 rport 50032 generation 0
a=ice-ufrag:fChVLLHDOF2MRiVr
a=ice-pwd:+1rKCJjEUfSRINYz15Flsi7I
a=ice-options:google-ice
a=fingerprint:sha-256
B4:5F:4D:B3:4A:0C:89:11:EB:48:9A:8D:93:94:35:A7:90:E1:B4:6C:E1:85:7E:04:7D:D5:EA:0F:1E:C3:AC:
a=setup:actpass
a=mid:video
a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=rtcp-mux
a=rtcp-fb:100 ccm fir
a=rtcp-fb:100 nack
a=rtcp-fb:100 nack pli
a=rtcp-fb:100 goog-remb
a=ssrc:781764365 cname:UYec8Hvix40P60XM
a=ssrc:781764365 msid:tiPYTIjN4dBCPJPy0WUo6hoV4cOIuubqww15
4a9ea9f2-e8cf-48b2-ae90-45232fe120d5
a=ssrc:781764365 mslabel:tiPYTIjN4dBCPJPy0WUo6hoV4cOIuubqww15
a=ssrc:781764365 label:4a9ea9f2-e8cf-48b2-ae90-45232fe120d5

音频编码支持：opus，万能经典的 G711，和 ISAC
视频编码支持：VP8，red 和 ulpfec

我只知道 VP8 的视频，其他两种没有听说.

火狐浏览器的 SDP，看起来少一些：

```
v=0
o=Mozilla-SIPUA-31.0 1555 0 IN IP4 0.0.0.0
s=SIP Call
t=0 0
a=ice-ufrag:d8cce358
a=ice-pwd:6ec490a6c7b333434728762554a9f94a
a=fingerprint:sha-256 06:C3:E7:23:1C:1C:6A:9B:E1:9D:42:EE:A5:BC:4E:1F:1B:2A:42:FD:CE:EC:52:C6:9E:3
m=audio 47520 RTP/SAVPF 109 0 8 101
c=IN IP4 211.162.33.95
a=rtpmap:109 opus/48000/2
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=ptime:20
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=setup:actpass
a=candidate:0 1 UDP 2128609535 192.168.1.5 56310 typ host
a=candidate:1 1 UDP 1692467199 211.162.33.95 47520 typ srflx raddr 192.168.1.5 rport 56310
a=candidate:0 2 UDP 2128609534 192.168.1.5 56311 typ host
a=rtcp-mux
m=video 47519 RTP/SAVPF 120
c=IN IP4 211.162.33.95
a=rtpmap:120 VP8/90000
a=rtcp-fb:120 nack
a=rtcp-fb:120 nack pli
a=rtcp-fb:120 ccm fir
a=setup:actpass
a=candidate:0 1 UDP 2128609535 192.168.1.5 56312 typ host
a=candidate:1 1 UDP 1692467199 211.162.33.95 47519 typ srflx raddr 192.168.1.5 rport 56312
a=candidate:0 2 UDP 2128609534 192.168.1.5 57165 typ host
a=candidate:1 2 UDP 1692467198 211.162.33.95 47518 typ srflx raddr 192.168.1.5 rport 57165
a=rtcp-mux
```

音频编码支持：opus，万能的经典的 G711。
视频编码支持：VP8

1.4.7 的 fs 默认音频编码是支持 opus，应该说默认是使用 opus 编码器。
Opus 语音编码器本人测试实际通话过程中平均总带宽大约是 50-60Kbps，声音质量也不错。G711 带宽到了 80 多 K，当然是优先考虑的 Opus 的。
Opus 在窄带的时候核心就是 Skype 的 SILK 编码。

opus 编码器 CPU 开销有点大，这个实施的时候需要注意，
但是使用 Opus 编码器的时候可能会有其他问题，这个下一个问题马上会提到。
iSAC 测试带宽 40Kbps 左右，问题是火狐不支持 iSAC
1.4.7 的 fs 默认视频编码没有，可以参照前面的视频问题，配置 H264 编码器或者 VP8 编码器。配置上去了才能支持视频。
由于 SDP 是浏览器内置的，貌似用户无法修改，因此只能适应他们的语音和视频编码。
因此总结如下：WebRTC 目前情况下使用 Opus 或者 G711 音频编码，使用 VP8 视频编码。
根据视频的大小，30 frames 情况
720P 1.0-2.0Mbps 大小 1280*720
360P 0.5-1.0Mbps 大小 480×360
视频 480*320 情况下带宽大约带宽 500-1000Kbps
由于两种浏览器都支持 VP8 视频.支持 VP8 视频的软电话我所知道的目前只有 IMSDroid。
媒体都使用 srtp 加密，对抓包跟踪比较麻烦。

236. FreeSwitch 如何集成 WebRTC 到系统中？

上面说过，FS 作为 WebRTC 的服务端，火狐或者 Chrome 作为客户端。
通常的使用场景浏览器客户端 WEB Phone 以 sip 分机注册到 FS 上，然后拨打其他注册上来的分机(包括 sip 话机，sip 软电话，flashphone，WEB Phone)，或者通过网关落地拨打到手机或者固话。
由于 FS 目前暂时不支持视频转码，
在仅仅考虑音频，没有视频的情况下。
列举几个 WAN 环境下最常用的使用场景：
A)WEB Phone == **è** WEB Phone
B)WEB Phone == **è** SIP 软电话比如 IMSDroid,eyebeam 或者 microsip
C)WEB Phone == **è** 落地网关到手机
D) SIP 软电话== **è** WEB Phone

场景 A 的情况，拨入是 Opus 编码器，拨出也是 Opus，测试效果没有问题。
场景 B 的情况，拨入是 Opus 编码器，拨出现在一般的软电话都是支持
G711，iLBC,G729，这些转码测试效果没有问题。
场景 C 的情况，拨入是 Opus 编码器，拨出现在一般的落地都是支持 G711，和 G729，这些转码测试效果没有问题。
场景 D 的情况，拨入是软电话都是支持 G711，iLBC,G729，然后拨出
是 Opus 编码器，问题来了，FS 会提示无法 DECODE。

freeswitch@AY1303090755553Z>
2014-09-11 20:04:04.513585 [ERR] switch_core_io.c:1283 Codec OPUS (STANDARD) decoder error!

chrome 作为被叫的时候 opus 编解码有问题，只能是 主叫也是 opus 的呼叫，

假如主叫是 alaw 呼叫 chrome 作为被叫的时候 opus 编解码有问题，chrome 只能选择 alaw，或者系统先呼叫通 chrome 然后再通过 uuid_bridge 实现通话
但是火狐作为被叫 microsip alaw 直接呼叫 opus 编解码没有问题
因此这种场景下，FS 不能配置 Opus 编码器，只能都配置 G711.

或者你自己花时间去修改那个不能转码的 bug.
另外 A 和 D 的场景，WEB Phone 作为被叫的情况下，假如是火狐浏览器，由于他的 STUN 包只会发几个，而在电话没有接通之前 fs 不会处理这些 STUN 包，导致电话接通之后
火狐不发包了。而 FS 没有收到包也不会回复到对应的端口上，这样 WAN 的环境下会有问题，这种场景下，需要设置 WEB Phone 为自动应答。才能完美实现。假如是 Chrome 会有问题，因为通话过程中它也会发 STUN 包。
WAN 环境下注意下面几点：
firefox 作为被叫不行， 比如 firefox2firefox 和 chrome2 firefox 都不行。
firefox 在作为被叫，应答之前发了 4 个 STUN 包，估计被 fs 抛弃，导致 NAT 穿透无效。
作为主叫是在对方应答之后才发 STUN 包，fs 有正常回应。NAT 穿透 OK
解决办法就是我上面说的需要设置 WEB Phone 为自动应答。

在有视频的情况下，软电话需要使用 IMSDroid 才可以视频。

237. FreeSwitch 如何修改才能实现以媒体代理方式支持 WebRTC？

上面说过，WebRTC 默认是 P2P 的，媒体不经过 FS，而且使用 ICE 作为 NAT 穿透。
要命的是 NAT 穿透就算是使用 ICE 也是经常不灵光，导致反而不能使用。
我的办法是让 FS 作为媒体代理，媒体都经过 FS，使用 FS 牛逼的 NAT 穿透办法（哪里去 BY 哪里来）来实现。
但是 FS 默认是 wan.auto 的需要改为 localnet.auto.
FS 里面需要修改代码才能实现这个功能：
修改
switch_core_media.c

```
if (!engine->cand_acl_count) {
        engine->cand_acl[engine->cand_acl_count++] = "localnet.auto";//wan.auto";yhy 2014-07-26
                    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(smh->session), SWITCH_LOG_WARNING,
"NO candidate ACL defined, Defaulting to yhy localnet.auto not wan.auto\n");
                }
```

238. FreeSwitch 如何修改才能完美支持火狐？

实际测试过程中发现火狐浏览器通话 60 秒左右自动没有声音 经过排查是重新发的 ice 导致的问提。
上面 的问题修改了媒体代理，但是火狐的每 60s 的 state 消息 会导致重新 ice 匹配导致媒体不行。
FS 里面需要修改代码才能实现完美支持火狐：
修改
sofia.c

```
if (r_sdp) {
                switch_channel_set_variable(channel, SWITCH_R_SDP_VARIABLE, r_sdp);

                if (!(profile->mndlb & SM_NDLB_ALLOW_NONDUP_SDP) || (!zstr(tech_pvt->mparams.remote_sdp_str)
&& !strcmp(tech_pvt->mparams.remote_sdp_str, r_sdp))) {
                    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(session), SWITCH_LOG_DEBUG,
"Duplicate SDP\n%s\n", r_sdp);
                    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(session), SWITCH_LOG_WARNING,
"Duplicate SDP:%s\n", "yhy skip");
                    //is_dup_sdp = 1;//yhy2014-08-08
                    goto done;
                } else {
                    switch_log_printf(SWITCH_CHANNEL_SESSION_LOG(session), SWITCH_LOG_DEBUG, "Remote
SDP:\n%s\n", r_sdp);
                    tech_pvt->mparams.remote_sdp_str = switch_core_session_strdup(session, r_sdp);

                  //if ((sofia_test_flag(tech_pvt, TFLAG_LATE_NEGOTIATION) || switch_channel_direction(channel) ==
SWITCH_CALL_DIRECTION_OUTBOUND)) {
                //      switch_core_media_set_sdp_codec_string(session, r_sdp, status < 200 ? SDP_TYPE_REQUEST :
SDP_TYPE_RESPONSE);
                    //}

                    sofia_glue_pass_sdp(tech_pvt, (char *) r_sdp);
                    sofia_set_flag(tech_pvt, TFLAG_NEW_SDP);

                }
            }
```

239. FreeSwitch+WebRTC 可以设计哪些业务？

针对 FS 不支持视频转码的特点。

我认为下面这些业务可以使用 WEB Phone 作为客户端：

A) 视频培训。
学员用户使用火狐或者 Chrome 浏览器，输入分配给他们的帐号密码
登录系统。
首页上最好要提供 chrome 和火狐绿色版本的下载，假如学员
使用 IE 浏览器则弹出提示 IE 暂时不支持，提示他们可以下载这两个浏览器来使用。
演示界面如下：

选择培训课程，然后观看老师的培训录像。
或者观看老师的实时培训，多人观看需要使用上面说的那个主持人控制视频会议的
办法来管理会场。
默认大家都只能看老师，假如老师有提问，老师可以控制切换，可以切换到发言的学员，这样大家都可以看到学生发言的视频情况。
演示界面大概如下

在 WAN 环境下使用 WEB Phone 参加 FS 的视频会议的使用有一个需要注意的地方：
经过上面不自动根据声音大小切换的修改修改之后。
发现第一个拨入可以看到自己视频，第二个拨入就看不到对方。
拨入之后先执行 echo，完成 nat 穿透， 3 秒之后
再进入会场，否则可能存在看不到会场里面的人的情况。

B) WEB Phone 呼叫中心。
使用 flash 插件作为坐席控制的数据控制通道，使用 WEB Phone 作为坐席的媒体通道。
客服使用使用火狐或者 Chrome 浏览器，输入分配给他们的帐号密码
登录系统，等待用户来电。用户的来电信息通过 flash 插件弹屏显示。
就是以
WEB Phone 代替一般的浏览器插件或者软电话作为坐席媒体的方式。
由于不需要额外安装插件或者额外的软电话，坐席可以随时随地上网提供服务。
而不需要固定的机器。
对于那种远程坐席，或者远程营销坐席比较有用。

C) WEB Phone 视频呼叫中心。
跟 B 类似，只是多了视频功能。
依然是使用 flash 插件作为坐席控制的数据控制通道，使用 WEB Phone 作为坐席的媒体通道。

客服使用使用火狐或者 Chrome 浏览器，输入分配给他们的帐号密码
登录系统，等待用户来电。用户的来电信息通过 flash 插件弹屏显示。
界面大概如下

上半部分是坐席控制的 flash 部分，
下半部分是 WEB Phone 部分。该页面测试拨打的是 1005 的另外一个分机。

第十五章　空号检测模块部分

240. 空号检测模块做啥用？

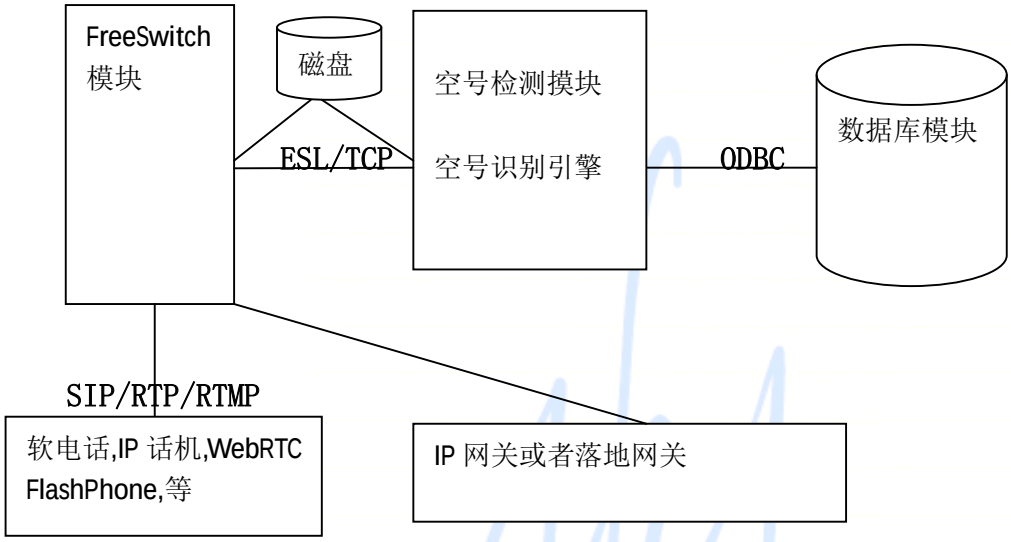
先讨论啥是空号检测模块，空号检测就是在在系统呼叫的时候判断某个被呼叫的号码是否异常，假如异常是否需要马上挂机。
在看看啥是呼叫异常，被呼叫号码的呼叫异常就是指系统拨打，被叫没有振铃，通常情况包括：
被叫是空号
被叫号码已经欠费停机
被叫号码已经关机
被叫号码正在通话中
被叫号码不在服务区
被叫号码转移到语音助理
。。。。
一句话，只要被叫没有振铃都算异常。

再讨论系统拨出的呼叫，一般的电话营销系统的呼叫有两种，一种是人工选号码拨出，一种是系统自动发起呼叫。人工拨出在系统收到 183 之后可以 link，人工坐席就可以听见线路上的声音。这个是不需要空号检测的。但是这种情况效率非常低。一个坐席一天呼叫不了几个。更加不用说成单了。这种情况比较时候呼叫中心的客户回访或者回复的情况，不太适合营销系统。
目前的营销系统通常都配置的自动外拨的群呼模块，
但是在系统自动形式外拨的时候，比如群呼的时候，目前的运营商环境，在用户无法接通（手机不振铃）的情况下是转录音通知。
有些运营商的交换机在 7 号的 ACM 信令上有说明原因，大多数运营商的交换机没有给出原因，7 号的网关都会认为用户是空闲，FreeSwitch 也当作正常的空闲返回 183 信令，系统会以为用户开始振铃，因此系统就等呀等呀，一直等到运营商的交换机播放 n 次录音通知之后发送 REL 拆线。然后系统才拆线，通常这个时间大约是 50-60 秒。因此白白浪费了中继的占用。假如有空号检测模块，在振铃之后开始录音，然后分析语音，通常只要 6-12 秒的时间就知道用户是啥情况。就可以马上拆线，让出中继资源，后续可以在该中继线路上呼叫其他电话。
再讨论号码，由于群呼的号码来源有两种:一种是市场上买的，号码资源好一些，停机，空号的少一些，接通率高一些，但是由于购买的号码的及时性无法保证，市场上买的也存在号码是停机和空号的情况。另外由于在用户忙，用户关机，用户不在服务区等情况下存在上面的浪费情况。
另外一种是直接呼叫某个号码比如从 13606060000 呼叫到 13606069999，这种呼叫大约 60-70%的号码不是停机就是空号，加上用户的用户忙，用户关机，用户不在服务区等各种异常情况，接通率就非常低。
总之，在自动外呼和群呼的系统里，配置空号检测模块的可以提高中继利用率达 1 倍以上。对系统来说必须的。

241. 基于 FreeSwitch 的空号检测模块如何设计？

系统基于 WINDOWS 操作系统，支持 windows XP，2003，2008，WIN7 等。
空号检测的结果通常需要写到用户的业务数据库通知用户呼叫的结果，.
系统支持使用 ORACLE/SQLSERVER/SYBASE/MYSQL 作为业务数据库。
空号检测模块支持下面 3 总使用模式

一. 单独录音版的空号检测模块体系结构如下：

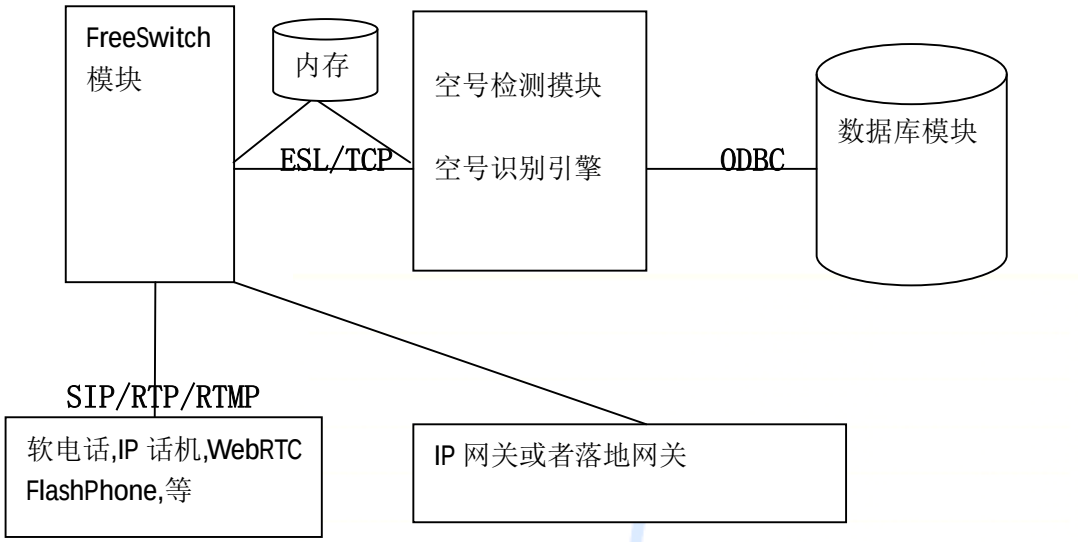


说明:

1. 空号检测模块通过 ESL 内联方式连接到 FreeSwitch，然后设置监听振铃的事件。
2. 在 FreeSwitch 外拨 ip 网关或者落地电话振铃的时候，空号检测模块获取到 UUID XXXX 开始启动录音。录音时间设置在 6-12 秒之间。
3. 空号检测模块录音结束，开始对录音的内容进行识别。
4. 假如识别结果是空号或者停机，或者关机，或者正在使用中，暂时无法接通等其他异常。则通过 ESL 给 FreeSwitch 发 uuid_kill xxxx 命令马上挂断该呼叫。假如是：振铃，彩铃和没有声音等其他情况当作正常处理。正常情况下不做任何处理。
5. 空号检测模块把处理记录写到数据库的 CDR 中。 当作呼叫结果的一部分。

优点：简单。业务和 FreeSwitch 不需要任何改动。通过配置空号检测模块就可以大大提高外呼，尤其是群呼的效率。
缺点：外拨振铃就开始录音。增加了 FreeSwitch 机器的磁盘 IO。

二. 单独共享内存版本的空号检测模块体系结构如下：



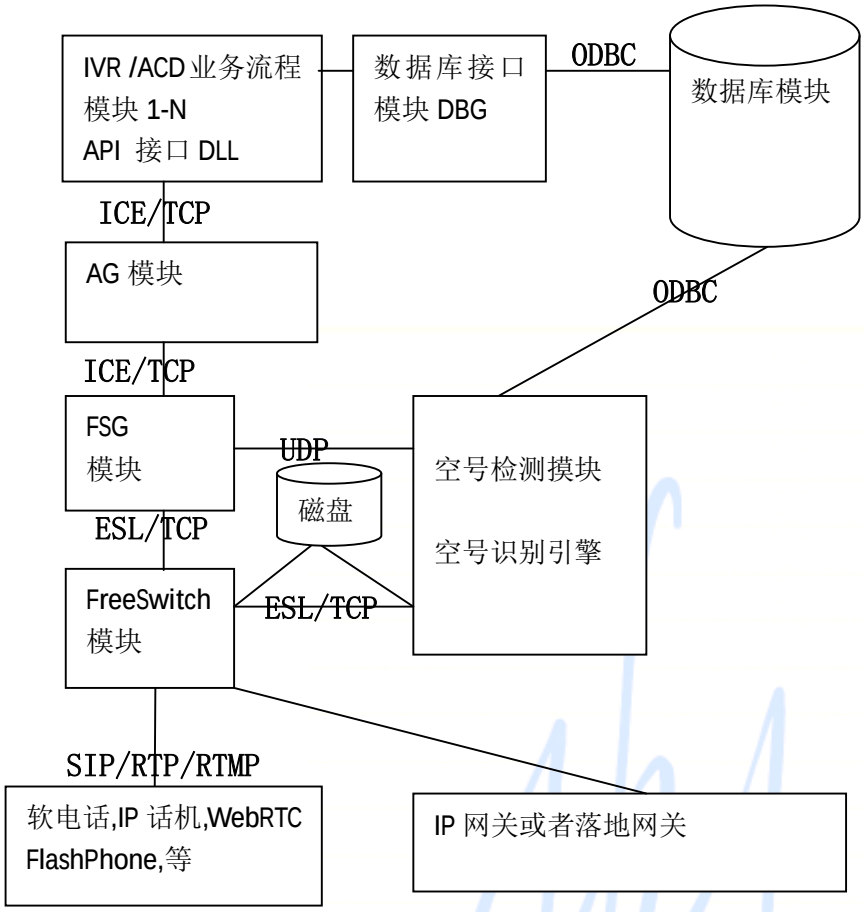
说明:

1. 空号检测模块通过 ESL 内联方式连接到 FreeSwitch，然后设置监听振铃的事件。
2. 在 FreeSwitch 外拨 ip 网关或者落地电话振铃的时候，空号检测模块获取到 UUID XXXX 开始启动录音，录音到共享内存里面。录音时间设置在 6-12 秒之间。
3. 空号检测模块定时取出内存里面的数据，进行识别，一旦识别出结果。马上停止录音
4. 假如识别结果是空号或者停机，或者关机，或者正在使用中，暂时无法接通等其他异常。则通过 ESL 给 FreeSwitch 发 uuid_kill xxxx 命令马上挂断该呼叫。假如是：振铃，彩铃和没有声音等其他情况当作正常处理。正常情况下不做任何处理。
5. 空号检测模块把处理记录写到数据库的 CDR 中。 当作呼叫结果的一部分。

优点：效率最高，时效最强，可以作为洗号系统的空号检测模块。通过配置空号检测模块就可以大大提高外呼，尤其是群呼的效率。

缺点：需要修改 FreeSwitch，本人是通过修改 mod_pocketsphinx 来实现的。实现比较复杂，因为要准实时检测开销也比较大。

三. 基于 FreeSwitch 的 CTI 开发平台的空号检测模块体系结构如下：



说明:

1. FSG 模块 在外拨 ip 网关或者落地电话振铃的时候发 UDP 消息通知 空号检测模块，空号检测模块开始启动录音。录音时间设置在 6-12 秒之间。UDP 的内容包括通话的 UUID。
2. 空号检测模块录音结束，开始对录音的内容进行识别。
3. 假如识别结果是空号或者停机，或者关机，或者正在使用中，暂时无法接通等其他异常。则通过 ESL 给 FreeSwitch 发 uuid_kill xxxx 命令马上挂断该呼叫。假如是：振铃，彩铃和没有声音等其他情况当作正常处理。正常情况下不做任何处理。
4. 空号检测模块把处理记录写到数据库的 CDR 中。 当作呼叫结果的一部分。

优点：FreeSwitch 的 CTI 开发平台完美结合，简单。业务和 FreeSwitch 不需要任何改动。通过配置空号检测模块就可以大大提高外呼，尤其是群呼的效率。

缺点：外拨振铃就开始录音。增加了 FreeSwitch 机器的磁盘 IO。

242. 基于 FreeSwitch 的空号检测模块核心引擎是什么？

空号检测模块核心引擎是原音搜索引擎，
该引擎是中科院声学所中科信利国家重点语音实验室的李明博士开发的，其核心思想是模式匹配。
该引擎最早是在 2005-2006 年开发的，那个时候彩铃开始流行。
相应的需求就是彩铃搜索，最初开发该引擎的目的是用来根据某段个音乐或者录音比对找出某是哪首歌曲。该引擎设计只要 2-3s 的有效语音就可以识别出是哪首歌曲。识别率在 99%以上。
从处理性能来说，该识别引擎使用 hash 查找，占用 CPU 很小。
单核处理器， 10 秒的语音比对识别只要 20ms。其处理语音可达到 100 倍实时以上。可以说是非常轻量级别的模式匹配引擎。使用到 FreeSwitch 的系统中做空号识别可以说非常合适。因为群呼的呼叫量特别大，假如引擎占用很多 CPU 就会影响到 FreeSwitch。
我把该引擎使用来比对交换机的录音通知，可以说的上是杀鸡用牛刀。
大家知道歌曲库是上万级别，而交换机的录音通知的个数跟歌曲库比较起来实在是小巫见大巫。
实网系统正常情况下使用空号检测的识别率在 99%以上。
使用该模块需要购买原音搜索引擎的授权才能使用，可以通过我或者之间找中科信利购买。

243. 基于 FreeSwitch 的空号检测模块语音库如何维护？

空号检测模块是使用模式匹配的技术来匹配语音的，模块自带了许多目前交换机的语音录音。
假如碰到某个交换机的通知录音的语音比较特殊，系统无法匹配。
把该录音拷贝 到模块的 wav 目录下，然后重新初始化参数就可以投入使用。
注意：为了保证良好的检测率，wav 目录下的语音里面不能有振铃音之类的语音。因此拷贝之前最好先使用 cooledit 语音处理工具编辑处理一下，去掉里面的振铃音，忙音等噪音。
否则正常的振铃也会被当作异常了。

附录 1 参考资料：

参考资料：
FreeSWITCH 权威指南 杜金房，张令考 著 购买地址： http://item.jd.com/11472569.html
高性能浏览器网络（影印版） Ilya Grigorik 著购买地址：http://item.jd.com/11493427.html
http://wiki.FreeSwitch.org/wiki/Main_Page
http://www.FreeSwitch.org.cn/
http://www.ppcn.net/n1306c2.aspx
http://midcom-p2p.sourceforge.net/draft-ford-midcom-p2p-01.txt
http://www.ctiforum.com/train/intel/base/base01_002.htm

附录 2 ESL IVR 控制代码下载：

您可以从 ftp 42.120.20.89 匿名下载 main.cpp
具体操作如下：

```
C:\Users\yhy>ftp 42.120.20.89
连接到 42.120.20.89。
220 Serv-U FTP Server v6.4 for WinSock ready...
用户(42.120.20.89:(none)): anonymous
331 User name okay, please send complete E-mail address as
密码:
230 User logged in, proceed.
ftp> ls
200 PORT Command successful.
150 Opening ASCII mode data connection for /bin/ls.
main.cpp
test.mxml
226 Transfer complete.
ftp: 收到 21 字节，用时 0.00 秒 21000.00 千字节/秒。
ftp>
```

登录匿名用户之后 密码为空，之后 asc，然后 get main.cpp 就可以

附录 3 FLEX flashphone 代码下载：

您可以从 ftp 42.120.20.89 匿名下载 test.mxml 具体操作参见：附录 2

附录 4　FreeSwitch 已知 bug：

参考资料：

FreeSwitch1.2.1　使用 flash 的 rtmp_mod 功能会崩溃。FreeSwitch 运行环境 win7 6G men B950CPU

FreeSwitch1.2.3　启动崩溃，怀疑是软电话 eyebeam 没有关闭，自动的注册，注册的时候 FreeSwitch 正在初始化某些东西导致崩溃 FreeSwitch 运行环境 win7 6G men B950CPU

FreeSwitch1.2.5.3　启动崩溃，怀疑是软电话 eyebeam 没有关闭，自动的注册，注册的时候 FreeSwitch 正在初始化某些东西导致崩溃 FreeSwitch 运行环境 win7 6G men B950CPU

FreeSwitch1.2.3　使用 flash 测试 会出现 thread failure　FreeSwitch 运行环境 win7 2G 内存，CPU :T7200

FreeSwitch1.2.5.3 使用 flash 测试 会出现 [CRIT] switch_core_session.c:1644 Thread Failure! FreeSwitch 运行环境 win7 2G 内存，CPU :T7200

FreeSwitch1.2.5.3 一直 使用 lash 很快呼叫，被叫马上接通，然后主叫马上挂机，会崩溃, FreeSwitch 运行环境 XP+sp3 2G MEM CPU E3300

FreeSwitch1.2.5.3 以及之前的 windows 和 linux 版本，mod_rtmp 压力测试 30 路拨入都会崩溃

FreeSwitch 1.2.7　的版本 mod_rtmp 60 路压力测试拨入不会崩溃，

　　解决了 mod_rtmp 崩溃的问题，因此建议使用 1.2.7 或以上版本

FreeSwitch 1.2.10 的版本：

假如 ImsDroid 的 里面的 public Identiry 里面的 sip:1031@192.168.1.102

误写成 sip:1031@1192.168.1.102

其它的配置都是对的情况下，FreeSwitch 会一直循环输出：

2013-07-18 14:35:37.515322 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request

 from 192.168.1.105:42614

2013-07-18 14:35:37.515322 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request

 from 192.168.1.105:42614

2013-07-18 14:35:37.515322 [ERR] sofia_reg.c:1161 Can not do authorization without a complete header in REGISTER request

 from 192.168.1.105:42614。。。。

貌似进入死循环，无穷无尽，马上导致其它功能受影响。这样很容易受到攻击。

FreeSwitch1.2.3–FS1.2.14, 所有版本，包括 1.4.0 的版本，使用分机对分机的呼叫进行测试，发现通话过程只要通话一直存在，而且没有说话，内存就会一直增加（泄漏），大约增加（泄漏）的速度是每线每秒 1K 左右，假如通话挂断，内存就释放。这样会导致在 32 位的操作系统上由于进程的内存不够用进而 FreeSwitch 进程崩溃。因此官方建议使用 64 位的操作系统。

FreeSwitch1.2.14-1.2.23

配置 odbc 连接 oracle,认证失败会内存泄漏，认证通过就不会泄漏。

FreeSwitc1.2.23

Mod_rtmp 会崩溃，在系统呼叫 flash phone 的时候比较容易重现。

FreeSwitc1.4.6

Debug 版本使用 WebRTC 的时候会崩溃，release 版本暂时未发现。